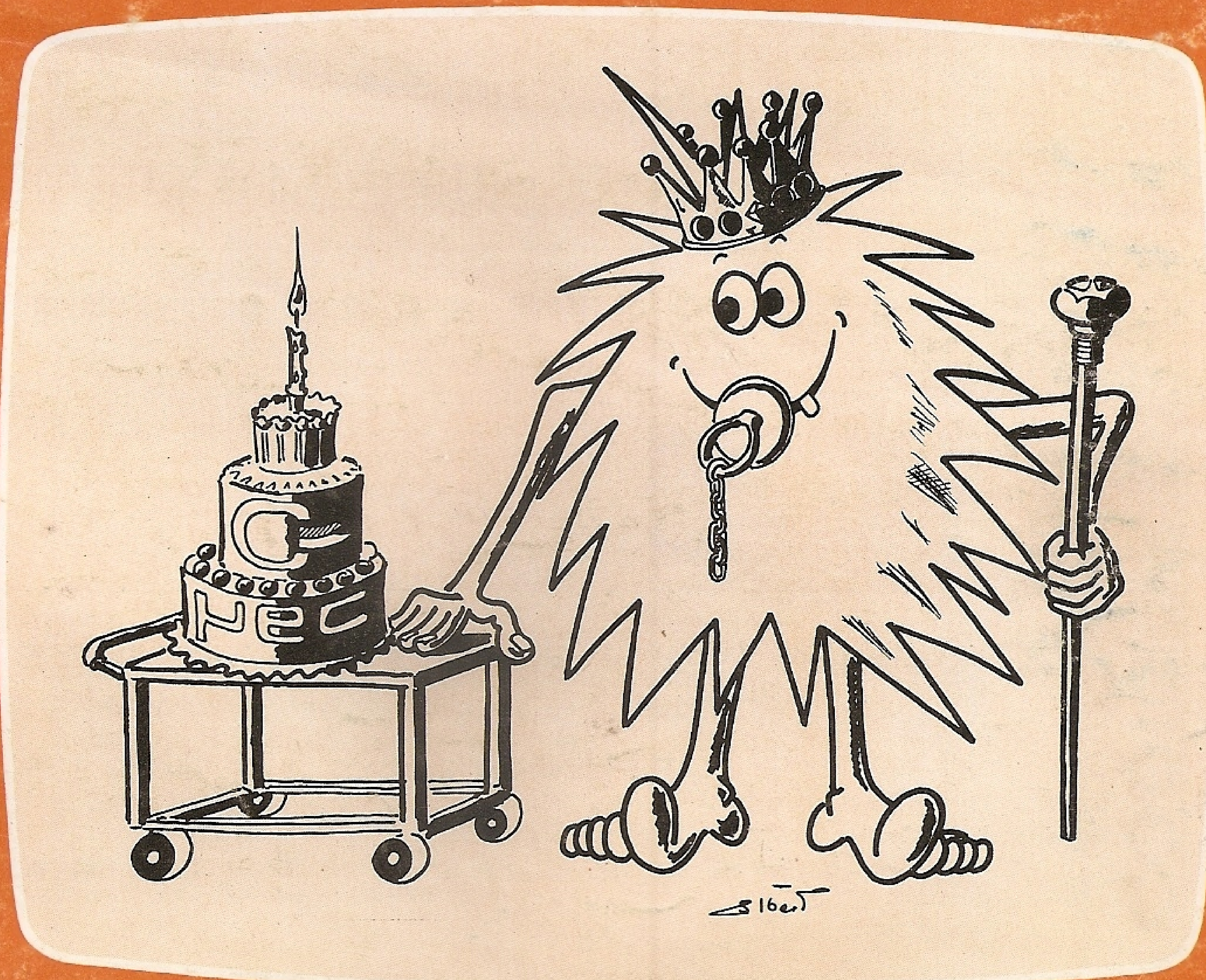
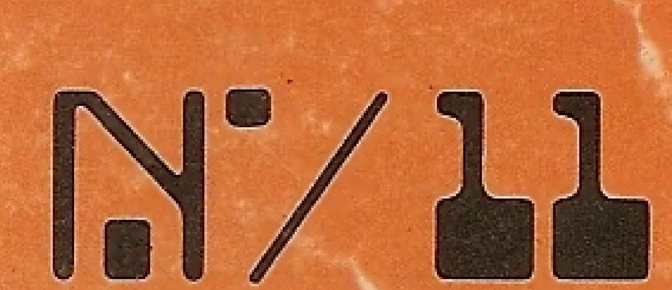
club commodore

Boletín informativo para los usuarios de microordenadores VIC y CBM

- un año –ya– de "CLUB COMMODORE" (pág. 1)
- ventana CBM: rutinas de cálculo del interpretador BASIC (pág. 1)
- software de base: funciones de manejo de fechas (pág. 5)
- código-máquina del 6502 (pág. 9)
- ficheros CBM: ficheros secuenciales (pág. 13)
- serie CBM 700 (I): los nuevos ordenadores de gestión serie 700 (pág. 17)





Nº/11 agosto/septiembre 1983

CÓDIGOS DE PANTALLA PARA EL COMMODORE 64

La lista siguiente facilita la tarea cuando se tiene que trabajar con POKEs en pantalla (posiciones de memoria 1024-2023). Se listan los dos conjuntos de caracteres (Mayúsculas y minúsculas) y el número con el que se deberá realizar el POKE. Como se sabe, los dos conjuntos no están disponibles al mismo tiempo, para cambiar manualmente de mayúsculas a minúsculas y viceversa se deben pulsar al mismo tiempo las teclas SHIFT y la tecla que tiene grabado el logotipo de COMMODORE. Para realizar la conmutación desde programa deben utilizarse los siguientes POKEs: POKE 53272,21 para pasar a Mayúsculas y POKE 53272,23 para pasar a minúsculas.

Todo número de la tabla puede exhibirse en video inverso, el código correspondiente se obtiene sumando 128 al valor de la lista.

JUEGO 1	JUEGO 2	POKE
@		0
A	a	1
B	b	2
C	c	3
D	d	4
E	e	5
F	f	6
G	g	7
H	h	8
I	i	9
J	j	10
K	k	11
L	l	12
M	m	13
N	n	14
O	o	15
P	p	16
Q	q	17
R	r	18
S	s	19
T	t	20
U	u	21
V	v	22
W	w	23
X	x	24
Y	y	25
Z	z	26
[		27
£		28
]		29
↑		30
←		31
SPACE		32
!		33
"		34
#		35
$		36
%		37
&		38
'		39
(		40
)		41
*		42
+		43
,		44
–		45
.		46
/		47
0		48
1		49
2		50
3		51
4		52
5		53
6		54
7		55
8		56
9		57
:		58
;		59
<		60
=		61
>		62
?		63
		64
♠	A	65
	B	66
	C	67
	D	68
	E	69
	F	70
	G	71
	H	72
	I	73
	J	74
	K	75
	L	76
	M	77
	N	78
	O	79
	P	80
●	Q	81
	R	82
♥	S	83

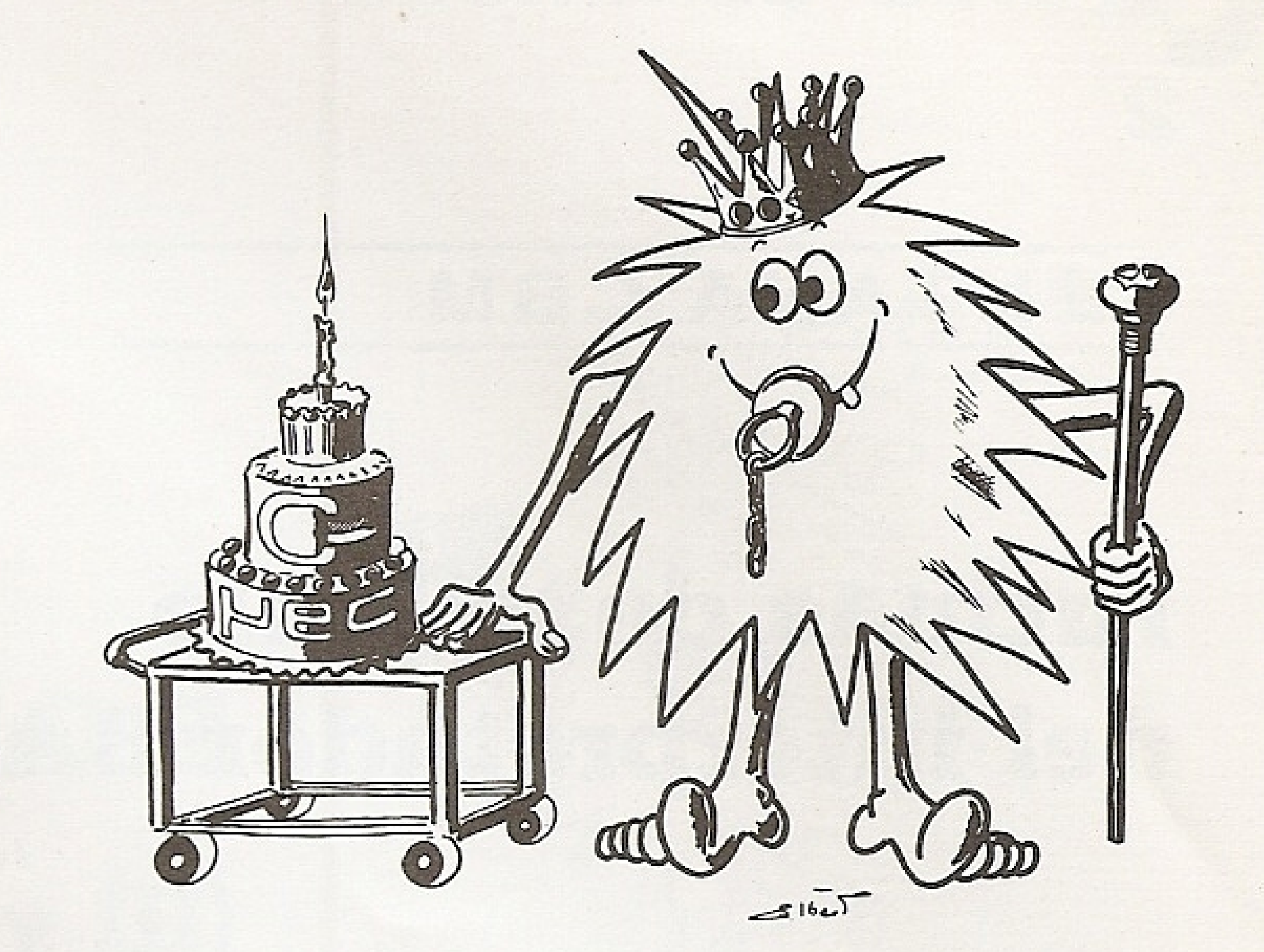
JUEGO 1	JUEGO 2	POKE
	T	84
	U	85
╳	V	86
○	W	87
♣	X	88
	Y	89
♦	Z	90
		91
		92
		93
		94
		95
SPACE		96
		97
		98
		99
		100
		101
		102
		103
		104
		105
		106
		107
		108
		109
		110
		111
		112
		113
		114
		115
		116
		117
		118
		119
		120
		121
	✓	122
		123
		124
		125
		126
		127

UN CORDIAL SALUDO A LOS VISITANTES DE SONIMAG-21

Desde las páginas de CLUB COMMODORE deseamos mandar un cordial saludo a los visitantes del certamen SONIMAG-21 que se celebrará en BARCELONA, del 26 de septiembre al 2 de octubre. Como en años anteriores MICROELECTRÓNICA Y CONTROL, S.A. estará presente con novedades en los productos que comercializa. Allí estará también CLUB COMMODORE. ¡Hasta pronto!

EDITORIAL

un año –ya– de "CLUB COMMODORE"



Suponemos al lector mentalmente preparado para recibir —después de leer el título de este editorial— una auténtica lluvia de frases del tipo: «Parece que fue ayer que empezábamos con esta Revista que pretende llenar un hueco en la biblioteca del aficionado a la microinformática y, sin darnos cuenta, ya hace un año que andamos dando la lata a este público que tanto nos quiere y al que tanto debemos, etc...»

Pero vamos a intentar —con la excusa del aniversario— ser un poco originales y repasar, muy por encima, lo que creemos que ha sido más importante para el desarrollo de la microinformática en nuestro país. En el último año han aparecido numerosas empresas dedicadas a la distribución de ordenadores personales y cierto número de publicaciones dedicadas a este tema, lo cual indica que, de una manera definitiva, el uso (y disfrute) de estos equipos está despegando —por fin— en España. Además, el número de usuarios ha crecido de una manera espectacular. Asimismo el nivel de conocimiento que hemos observado en los usuarios de ordenadores personales a los que hemos tenido el placer de conocer personalmente ha experimentado una sustancial mejora con claros indicios de continuar en la misma línea, mejora en la cual deseamos —y esperamos— haber tenido algo que ver.

Se está produciendo ahora un fenómeno del que ya habíamos oído hablar en otros países: consiste en encontrar a expertos programadores con edades inferiores a los trece años. Más de un programador de la vieja guardia anda ligeramente «acongojado» con esta circunstancia y la verdad es que los que somos mayorcitos deberemos espabilarnos si no queremos que la «nueva generación» nos siegue la hierba bajo los pies (que ha sido siempre la principal afición de las nuevas generaciones).

No queremos soltar el proceso de textos con el que redactamos este editorial sin aprovechar la ocasión para agradecer a REVISTA ESPAÑOLA DE ELECTRÓNICA la inestimable ayuda que nos ha prestado para la elaboración de nuestra Revista, sin la cual —no sólo no hubiera sido posible celebrar este primer aniversario— sino los muchos que tenemos intención de cumplir.

¡Hasta pronto! Y como siempre: ¡A ver si llegan más colaboraciones! ■

VENTANA CBM

rutinas de cálculo del interpretador BASIC

por **RAFAEL NAVARRO** (M.E.C. SOFT.)

Como os comenté en el número anterior, el interpretador de BASIC de COMMODORE utiliza, para el almacenamiento de números y resultados de operaciones, dos acumuladores. La longitud de los mismos es de seis bytes y están ambos situados en la página cero, siendo las posiciones ocupadas por el acumulador # 1 las $5E-$63 y las del acumulador # 2, las $66-$6B.

Aunque se esté trabajando con variables enteras o con expresiones numéricas literales, el interpretador necesita pasar a representación en coma flotante todas las expresiones para poder operar con ellas. De este modo, en algunas ocasiones, puede presentar una ventaja la utilización de variables de coma flotante incluso para valores enteros ya que tal práctica evitará tiempos de cambio de representación al interpretador.

Os presento a continuación una breve descripción de un grupo de rutinas muy interesantes que podréis utilizar desde código máquina para operar con valores. En la tabla adjunta os doy las direcciones de entrada para las tres versiones de BASIC. Recordad que el BASIC 1 es el del veterano PET 2001, el BASIC 2 es el del CBM 3032 y el del VIC-20 y el BASIC 4 corresponde a las series CBM 4032 y CBM 8032. Las direcciones de página cero encerradas entre paréntesis representan punteros indirectos, es decir, el dato se encuentra en la dirección calculada como PUNTERO + (PUNTERO + 1)* 256.

En próximos números comentaremos otras rutinas en las que están implicados los dos acumuladores.

(continúa en la pág. siguiente)

VENTANA CBM

rutinas de cálculo del interpretador BASIC

(viene de la pág. anterior)

TABLA DE RUTINAS MATEMÁTICAS DE "COMMODORE"

BASIC 1	BASIC 2	BASIC 4	DESCRIPCIÓN
$D6DA	$D6D2	$C92D	Convierte el acumulador flotante # 1 en un entero de dos bytes en ($11) y, con los pesos invertidos en ($61). A la salida, .A (el acumulador) contiene el peso alto y .Y (registro Y) el peso bajo.
$D71E	$D72C	$C97F	Suma 0.5 al acumulador # 1, dejando el resultado de la operación en el mismo.
$D8BF	$D8F6	$CB20	Convierte el acumulador # 1 en su logaritmo neperiano.
$D81C	$D853	$CA7D	Reemplaza el acumulador # 1 por su complemento a 2.
$D9B4	$D9EE	$CC18	Multiplica por 10.
$DAED	$DB27	$CD51	Redondea el acumulador # 1 utilizando el sexto byte (recordad que la representación de números en los acumuladores es un byte más largo que en las variables).
$DAFD	$DB37	$CD61	Determinación del signo. A la salida, .A = 0 si el acumulador # 1 = 0, 1 si es positivo, e = 255 si es negativo.
$DB2A	$DB64	$CD91	Compara el contenido del acumulador # 1 con la representación en 5 bytes en coma flotante apuntada por .A (peso bajo) e .Y (peso alto). A la salida, .A = 0 si son iguales, = 1 si el acumulador # 1 es mayor que el contenido de la memoria, e = 255 si es menor.
$DB6D	$DBA7	$CDD1	Pasa el contenido del acumulador # 1 a un valor entero en dos bytes $61 (alto) y $62 (bajo). Realiza un trabajo similar a la primera rutina de esta tabla, con la diferencia de que la primera imprime un mensaje ?ILLEGAL QUANTITY ERROR si el valor en el acumulador está fuera de rango y esta rutina no efectúa dicho control.
$DB9E	$DBD8	$CE02	Calcula la parte entera del acumulador # 1 dejando el resultado en representación de coma flotante en el mismo.
$DBBB	$DBF5	$CE1F	Coloca ceros binarios en el acumulador # 1 si el exponente es cero.
$DE67	$DEA1	$D14B	Cambia el signo del acumulador # 1.
$DFA5	$DFDF	$D289	Calcula el seno del acumulador # 1 considerando que el argumento se entrega en radianes.
$E048	$E08C	$D32C	Calcula el arcotangente del acumulador # 1.

■

Para terminar esta serie sobre los Ficheros Relativos, voy a volver sobre los temas que tratamos en el primer artículo (ver Club Commodore 9) pero más extensamente y con nuevos ejemplos.

CREACIÓN DE FICHEROS RELATIVOS

Lógicamente, antes de trabajar con un Fichero Relativo, éste debe crearse para reservar el espacio correspondiente en el disco. Para ello, basta con abrir el fichero, posicionarse sobre el último registro que se desea contenga el fichero, efectuar en él una grabación, y cerrarlo. Por ejemplo:

```
100 DOPEN #1,«FICHERO»,D0,L50
110 RECORD #1,100
120 PRINT #1
130 DCLOSE #1
```

En la instrucción DOPEN #, el parámetro L50 indica la longitud de los registros (50 bytes en este caso). En la creación del fichero es imprescindible poner este parámetro, si no se hiciera, el DOS emitiría un mensaje de 62,FILE NOT FOUND («fichero no encontrado»).

La instrucción RECORD # sitúa el puntero sobre el registro número 100, que todavía no existe. Por consiguiente, el DOS provoca el mensaje de error 50,RECORD NOT PRESENT («registro no existe»), que no debe tomarse como tal error, sino simplemente como un aviso.

La instrucción PRINT # provoca que el DOS cree el registro 100 y todos los anteriores, junto con los sectores-índice correspondientes. Si en esta instrucción se pone un dato a grabar, por ejemplo, PRINT #1,CHR$(65), el DOS graba este carácter al principio de todos los registros que cree. Si no se indica nada, como en el caso del ejemplo, asigna un CHR$(255) al primer byte de todos los registros creados.

Si mientras el fichero se está generando, se agota el espacio disponible de disco, el DOS genera un error 52, FILE TOO LARGE («fichero demasiado grande»).

La instrucción DCLOSE # provoca la actualización del BAM y del contador de bloques libres del directorio.

Una vez el fichero creado, ya se puede operar con él.

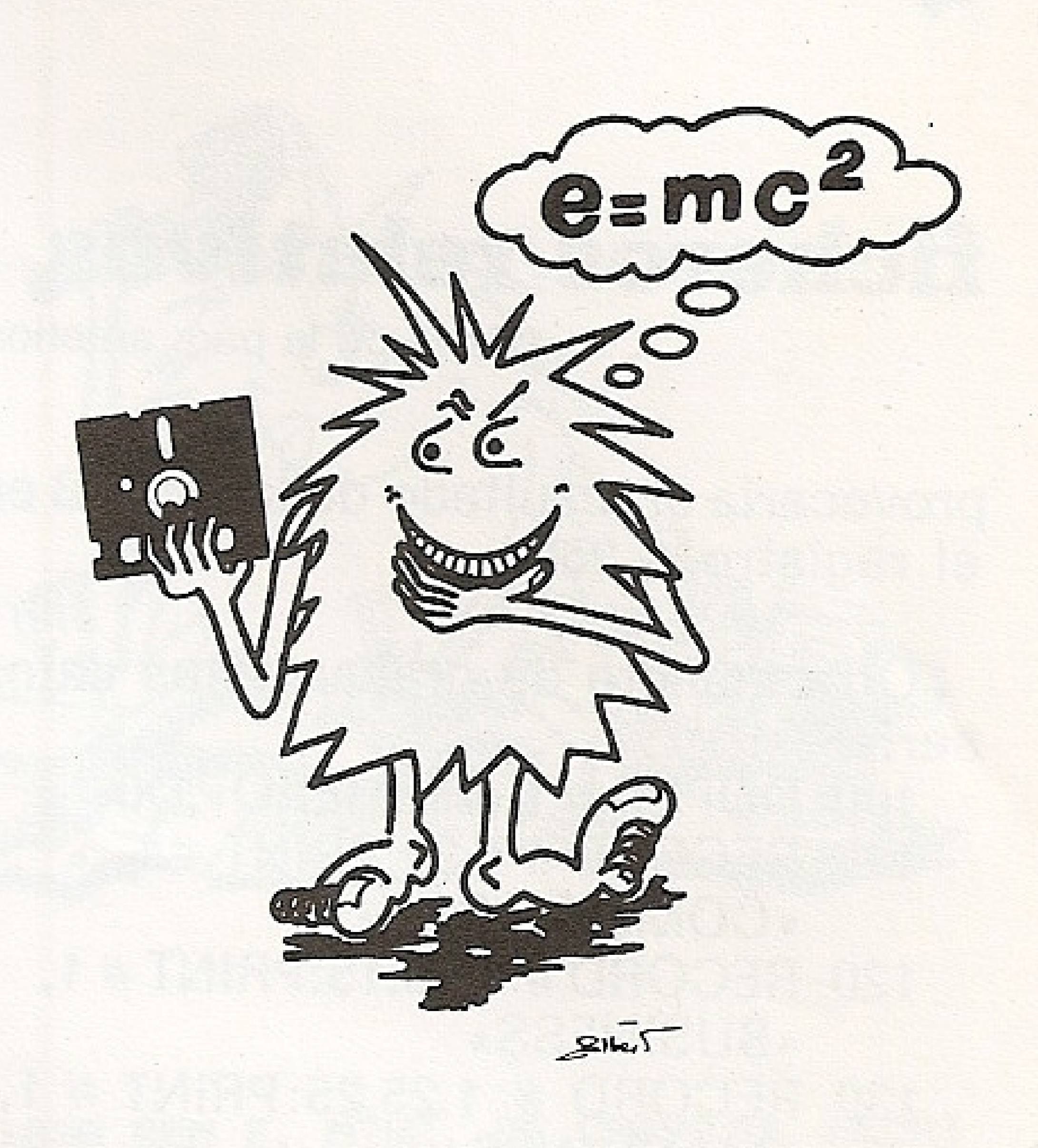
EXPANSIÓN DE FICHEROS RELATIVOS

Para expandir un Fichero Relativo, se usa el mismo procedimiento que

FICHEROS

ficheros relativos

(y III)



por **JORDI SASTRE**
(M.E.C. SOFT)

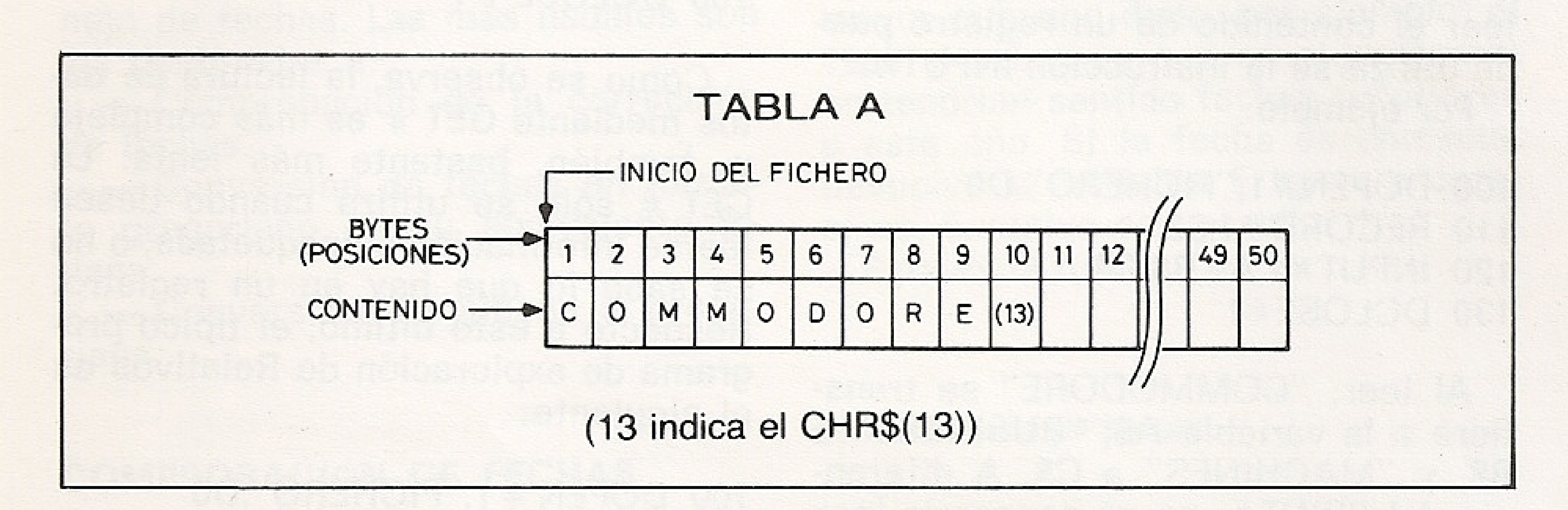


TABLA A

INICIO DEL FICHERO

BYTES (POSICIONES)	1	2	3	4	5	6	7	8	9	10	11	12		49	50
CONTENIDO	C	O	M	M	O	D	O	R	E	(13)					

(13 indica el CHR$(13))

para la creación pero, esta vez, posicionando sobre el nuevo número de registros que se desean.

El parámetro de la longitud del registro de la instrucción DOPEN# (L50) es opcional cuando el fichero ya está creado. Ahora bien, si se le indica una longitud que no coincide con la que tiene memorizada, el DOS emitirá un mensaje de error 50,RECORD NOT PRESENT.

Cuando el fichero es expandido de esta manera, también se crean los sectores-índice necesarios hasta llegar a la nueva longitud del fichero, pero recordemos que éstos son totalmente transparentes al programador, puesto que los gestiona automáticamente el DOS.

ACCESO A FICHEROS RELATIVOS

Previamente a cualquier lectura o grabación de datos sobre Ficheros Relativos, debe posicionarse el puntero sobre el registro deseado. Ello se hace mediante la instrucción RECORD# :

RECORD # nf , nr , nb

nf = número de fichero.
nr = número de registro dentro del fichero.
nb = número de byte dentro del registro.

El último parámetro es opcional, y su omisión indicará al DOS que la posterior operación con el registro (lectura o grabación) debe iniciarse al principio del registro.

Ejemplo de grabación en el fichero abierto anteriormente:

```
100 DOPEN #1,«FICHERO»,D0
110 RECORD #1,25
120 PRINT # 1,«COMMODORE»
130 DCLOSE #1
```

El DOS ha grabado en el registro # 25, y a partir del byte # 1, la palabra COMMODORE y a continuación un retorno de carro (carácter 13), que indica el final del campo (tabla A).

Para grabar dos o más campos en un mismo registro, debe utilizarse este mismo CHR$(13) como separador entre campos, por ejemplo:

```
100 DOPEN # 1,«FICHERO»,D0
110 RECORD #1,25
120 PRINT # 1,«COMMODORE»
    CHR$(13)«BUSINESS»CHR$
    (13)«MACHINES»
130 DCLOSE #1
```

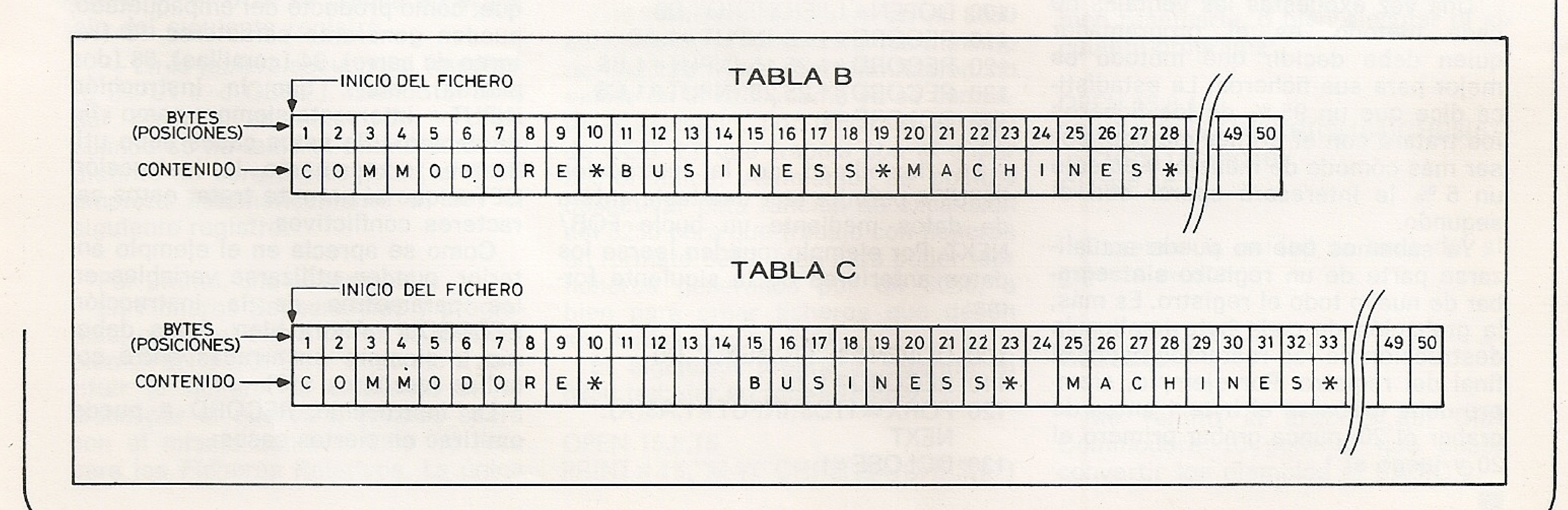


TABLA B

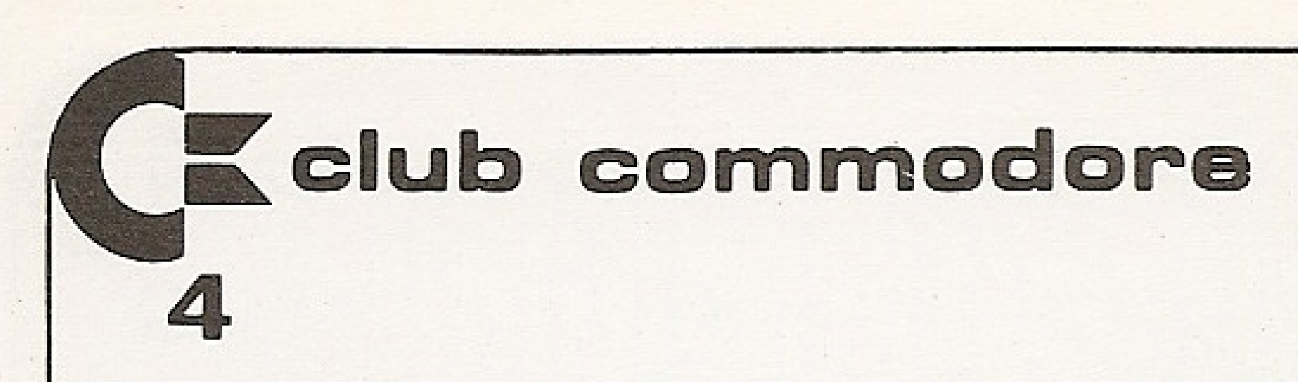
INICIO DEL FICHERO

BYTES (POSICIONES)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28		49	50
CONTENIDO	C	O	M	M	O	D	O	R	E	*	B	U	S	I	N	E	S	S	*	M	A	C	H	I	N	E	S	*			

TABLA C

INICIO DEL FICHERO

BYTES (POSICIONES)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33		49	50
CONTENIDO	C	O	M	M	O	D	O	R	E	*					B	U	S	I	N	E	S	S	*		M	A	C	H	I	N	E	S	*			



ficheros relativos

(viene de la pág. anterior)

provocaría el resultado de la tabla B en el registro # 25.

Otra manera de grabar estos datos sería:

```
100 DOPEN #1,«FICHERO»,D0
110 RECORD # 1,25:PRINT #1,
    «COMMODORE»
120 RECORD # 1,25,15:PRINT # 1,
    «BUSINESS»
130 RECORD # 1,25,25:PRINT # 1,
    «MACHINES»
140 DCLOSE # 1
```

que provocaría lo que se representa en la tabla C.

Estas dos maneras de grabación de datos precisan de una comparación exhaustiva, enumerando las ventajas de ambas:

Ventajas del primer método: No se desperdicia ningún byte entre los campos, excepto el CHR$(13) de separación. Se precisan de menos instrucciones BASIC para leer y grabar los campos (ver ejemplos). Debido a que no puede actualizarse únicamente un campo de un registro (al grabar siempre hay que grabar el registro entero), generalmente no tiene importancia dónde empieza cada campo ni qué longitud tenga, sólo importa el orden en que están grabados.

Por contrapartida, el segundo método también tiene su ventaja: se puede leer uno sólo de los campos prescindiendo de los demás porque se sabe dónde empieza. Ello sólo es válido para consultas, no para actualización de registros porque, al actualizar el campo hay que regrabar todo el registro, por lo que antes deben leerse todos los campos.

Una vez expuestas las ventajas de cada método, es el programador quien debe decidir qué método es mejor para sus ficheros. La estadística dice que un 95 % de los ficheros los tratará con el primer método, por ser más cómodo de manejo, y en sólo un 5 % le interesará operar con el segundo.

Ya sabemos que no puede actualizarse parte de un registro sin regrabar de nuevo todo el registro. Es más, la grabación de cada byte implica la destrucción de los restantes hasta el final del registro. Por ejemplo, siempre debe grabarse el byte 1 antes de grabar el 20, nunca grabar primero el 20 y luego el 1.

El DOS sólo acepta una instrucción PRINT # por cada instrucción RECORD #. En más de un caso el programador se encontrará con que el contenido a grabar en un registro no le cabe en una única instrucción PRINT#. La solución es poner primero todo el contenido del registro en una variable de cadena, y luego grabar ésta de una sola vez. Por ejemplo:

```
100 DOPEN#1,"FICHERO",D0
110 RECORD#1,25
120 B$="COMMODORE"+CHR$(13)
130 B$=B$+"BUSINESS"+CHR$
    (13)+"MACHINES"
140 PRINT#1,B$
150 DCLOSE#1
```

Puesto que el CHR$(13) actúa como separador entre campos, para leer el contenido de un registro puede utilizarse la instrucción INPUT#.

Por ejemplo:

```
100 DOPEN#1,"FICHERO",D0
110 RECORD#1,25
120 INPUT#1,A$,B$,C$
130 DCLOSE#1
```

Al leer, "COMMODORE" se transfiere a la variable A$, "BUSINESS" a B$, y "MACHINES" a C$. A diferencia del PRINT#, no es necesario leer todo el registro con una única instrucción INPUT#. Estos datos también hubieran podido leerse de la siguiente manera:

```
100 DOPEN#1,"'FICHERO",D0
110 RECORD#1,25
120 INPUT#1,A$
130 INPUT#1,B$
140 INPUT#1,C$
150 DCLOSE#1
```

el resultado es el mismo.

Si los datos han sido grabados utilizando el tercer parámetro de la instrucción RECORD#, deben leerse mediante el siguiente procedimiento:

```
100 DOPEN#1,"FICHERO",D0
110 RECORD#1,25:INPUT#1,A$
120 RECORD#1,25,15:INPUT#1,B$
130 RECORD#1,25,25:INPUT#1,C$
140 DCLOSE#1
```

La flexibilidad de la instrucción INPUT# permite leer una tabla entera de datos mediante un bucle FOR/NEXT. Por ejemplo, pueden leerse los datos anteriores de la siguiente forma:

```
100 DOPEN#1,"FICHERO",D0
110 RECORD#1,25
120 FORX=1TO3:INPUT#1,A$(X):
    NEXT
130 DCLOSE#1
```

En este caso, los datos se asignan a las variables A$(1)="COMMODORE", A$(2)="BUSINESS" y A$(3)="MACHINES".

También puede utilizarse la instrucción GET # para leer registros, por ejemplo, si la información está empaquetada. Un ejemplo de lectura de los anteriores datos utilizando la instrucción GET # sería:

```
100 DOPEN # 1,"FICHERO",D0
110 RECORD # 1,25
120 A$="":B$="":C$=""
130 GET # 1,X$ : IF X$<>CHR$(13)
    THEN A$=A$+X$ : GOTO130
140 GET # 1,X$ : IF X$<>CHR$(13)
    THEN B$=B$+X$ : GOTO140
150 GET # 1,X$ : IF X$<>CHR$(13)
    THEN C$=C$+X$ : GOTO150
160 DCLOSE #1
```

Como se observa, la lectura de datos mediante GET # es más compleja y, también, bastante más lenta. La GET # sólo se utiliza cuando desea leerse información empaquetada, o no se sabe lo que hay en un registro. Respecto a esto último, el típico programa de exploración de Relativos es el siguiente:

```
100 DOPEN # 1,"FICHERO",D0
110 RECORD # 1,(NR)
    :REM NR = Número de Registro
120 GET # 1,A$ : IF ST=2 THEN DCL
    OSE # 1 : END
130 PRINT A$;
140 GOTO 120
```

Si se prevé que el contenido del registro puedan ser caracteres de control o caracteres cuyo equivalente en ASCII no sea presentable en la pantalla, puede cambiarse la línea 130 y poner:

```
130 PRINT ASC(A$+CHR$(0));
```

Cuando se almacena información empaquetada en un registro, ésta no puede leerse mediante INPUT # porque, como producto del empaquetado, pueden generarse caracteres 13 (retorno de carro), 34 (comillas), 58 (dos puntos), etc... que la instrucción INPUT # interpreta siempre como «fin de campo». En estos casos debe utilizarse forzosamente la instrucción GET # que sí permite tratar estos caracteres conflictivos.

Como se aprecia en el ejemplo anterior, pueden utilizarse variables en los parámetros de la instrucción RECORD #. Ahora bien, éstas deberán ir siempre encerradas entre comillas (ver línea 110).

La instrucción RECORD # puede omitirse en ciertos casos:

SOFTWARE DE BASE (VI)

funciones de manejo de fechas

por E. MARTÍNEZ DE CARVAJAL

Algunas versiones de BASIC disponen de funciones que facilitan el manejo de fechas. Las más usuales son las destinadas a:

— Comprobación de la corrección de fechas.

— Conversión de fechas en formato gregoriano en su equivalente juliana.

— Conversión de juliana a gregoriana.

COMPROBACIÓN DE FECHAS

Esta función se utiliza para comprobar la corrección de una fecha entrada por el usuario en una sentencia INPUT. Las comprobaciones que se hacen son que el mes esté entre 1 y 12, que el día sea correcto, teniendo en cuenta el mes y si el año es bisiesto, y finalmente que el año sea mayor que uno dado, por ejemplo, el 1980 si sabemos que en el programa no tendrían sentido fechas anteriores a este año. Si la fecha es correcta, devuelve la variable E igual a cero; si no, E vuelve con el valor 1.

Ésta es la rutina:

```
PROGRAMA:11SOFTBASE1

1000 REM * CTRLDATE *
1010 REM ---------------------------------
1020 REM COMPROBACION FECHAS
1030 REM E. MTNZ. DE CARVAJAL HEDRICH
1040 REM 25/05/83
1050 REM ---------------------------------
1060 REM PARAMETROS DE ENTRADA : FE$ CON FORMATO DDMMAA
1080 REM PARAMETROS DE SALIDA  : E=0 (O.K), E=1 (ERROR)
1090 REM ---------------------------------
1100 DIM ND(12)
1110 DATA 31,28,31,30,31,30,31,31,30,31,30,31
1120 RESTORE
1130 FOR I=1 TO 12
```

(continúa en la pág. siguiente)

ficheros relativos (conclusión)

— Cuando se abre un fichero, automáticamente el DOS se posiciona sobre el primer byte del primer registro.

— Al término de cada instrucción PRINT #, el DOS se coloca en el inicio del siguiente registro.

— En la lectura secuencial (mediante INPUT # o GET #), cuando el final del registro es detectado, o sea, cuando se ha leído el último carácter distinto de cero del registro, el DOS empieza automáticamente a leer el siguiente registro.

Por último, una observación:

En Ficheros Secuenciales y programas, poner el signo @ en la instrucción OPEN o DOPEN # equivale a sustituir el fichero ya existente en el disco por el que va a crearse ahora con el mismo nombre. Esto no reza para los Ficheros Relativos. La única manera de borrar un Fichero Relativo es utilizando el comando SCRATCH.

COMPATIBILIDAD ENTRE EL 8050 Y EL 8250

Los Ficheros Relativos en el 8050 están limitados a un volumen máximo de 182.880 bytes. Esta limitación no existe en el 8250, como ya se comentó al principio, por llevar la versión de DOS 2.7 que trabaja con el super sector-índice. Asimismo, es posible desactivar esta función de expansión del 8250 para que trabaje con ficheros en formato 8050, bien para poder leer ficheros generados por un 8050, o bien para crear ficheros que deben ser leídos por un 8050.

El siguiente programa permite al 8250 trabajar en formato 8050:

```
OPEN 15,8,15
PRINT # 15,"M-W"CHR$(164)CHR$(67)
  CHR$(1)CHR$(255)
CLOSE 15
```

Para restaurar el 8250 en sus funciones normales, basta con desconectar y conectar de nuevo la unidad, o bien resetearla, o bien ejecutar el siguiente programa:

```
OPEN 15,8,15
PRINT # 15,"M-W"CHR$(164)CHR$(67)
  CHR$(1)CHR$(0)
CLOSE 15
```

Resetear la unidad se hace del siguiente modo:

```
OPEN 15,8,15,"U:"
CLOSE15
```

Me remito al artículo del Club Commodore 10 para el que desee convertir los ejemplos al BASIC 2.

funciones de manejo de fechas

(viene de la pág. anterior)

```
1140 READ ND(I)
1150 NEXT I
1160 IF LEN(FE$)<>6THENGOTO 1310
1170 E=0 : DI=VAL(LEFT$(FE$,2))
1190 ME=VAL(MID$(FE$,3,2))
1210 AN=VAL(RIGHT$(FE$,2))
1230 IF AN<80 THEN 1310
1240 ND(2)=28
1250 X1=AN:X2=4:GOSUB9000:IFX4=0THE
NND(2)=29
1260 IF ME<1 THEN 1310
1270 IF ME>12 THEN 1310
1280 IF DI<1 THEN 1310
1290 IF DI>ND(ME) THEN 1310
1300 RETURN
1310 REM ** ERROR **
1320 E=1 : RETURN
9000 REM MOD
9010 REM
9020 REM DATOS DE ENTRADA :
9030 REM
9040 REM X1=DIVIDENDO
9050 REM X2=DIVISOR
9060 REM
9070 REM DATOS DE SALIDA
9080 REM
9090 REM X3=PARTE FRACCIONARIA
9100 REM X4=RESTO
9110 REM
9120 X3=INT(X1/X2)
9130 X4=X1-X3*X2
9140 X3=(X1/X2)-X3
9150 RETURN
READY.
```

CONVERSIÓN DE GREGORIANA A JULIANA

Antiguamente, hasta el año 1572 aproximadamente, las fechas se utilizaban como días transcurridos desde la proclamación de Julio César como emperador (fechas julianas), con años de 365 días y cada tres uno bisiesto de 366. Sin embargo, esto no era correcto y, ya en su día, el emperador Augusto lo corrigió a un año bisiesto cada cuatro. A pesar de esto, como el año trópico o real es once minutos menor que el juliano, resultó que cada 130 años se atrasaban un día. Así se encontraron por el siglo XV que llevaban un adelanto de 10 días respecto de la fecha que utilizaban. Fue entonces cuando el Papa Gregorio XIII propuso la reforma del calendario, creando el que en la actualidad utilizamos (fechas gregorianas), y adelantando la fecha en los diez días que le faltaban.

A pesar de tanta reforma, lo cierto es que en informática, por curioso que parezca, es más útil manejar las fechas como días transcurridos desde una dada, es decir, en la forma inicial o juliana, sobre todo para hacer comparaciones entre ellas, para calcular vencimientos, etc...

Las dos siguientes rutinas permiten la conversión de una fecha gregoriana con formato dd/mm/aaaa en su juliana tomando como base el 1 de enero de 1980, y la inversa, que transforma una fecha juliana en su equivalente gregoriana.

En la primera de las rutinas, se utiliza la función POS (position) que ya os definí en un artículo anterior.

```
PROGRAMA:11SOFTBASE2

READY.

1000 REM JULIANA
1010 REM CONVERSION GREGORIANA A JU
LIANA
1020 REM ---------------------------
-------
1030 REM ERNESTO MTNZ. DE CARVAJAL
HEDRICH
1040 REM 25/05/83
1050 REM ---------------------------
-------
1060 REM PARAMETROS DE ENTRADA : FE
CHA EN FORMATO DD/MM/AAAA EN VARIAB
LE FE$
1070 REM PARAMETROS DE SALIDA  : FE
CHA EN JULIANA EN VARIABLE JU
1080 REM ---------------------------
-------
1090 DIM ND(12)
1100 DATA 31,28,31,30,31,30,31,31,3
0,31,30,31
1110 FOR I=1 TO 12:READ ND(I):NEXT
I
1120 X1$=FE$:X2$="/":X1=1:GOSUB 129
0:REM POS
1130 DI=VAL(LEFT$(FE$,X2-1))
1140 X3=X2+1:X1=X3:GOSUB 1290:REM P
OS
1150 ME=VAL(MID$(FE$,X3,(X2-X3)))
1160 AN=VAL(RIGHT$(FE$,4))
1170 JU=(AN-1980)*365
1180 IF ME=1 THEN 1200
1190 FOR I=1 TO ME-1:JU=JU+ND(I):NE
XT I
1200 JU=JU+DI
1210 IF AN-1980=0THEN1260
1220 FOR I=1980 TO AN-1
1230 X1=I:X2=100:GOSUB9000:IFX3=0TH
EN1250
1240 X1=I:X2=4:GOSUB9000:IFX3=0THEN
JU=JU+1
1250 NEXT I
1260 X1=AN:X2=100:GOSUB9000:IFX3=0T
HEN1280
1270 X1=AN:X2=4:GOSUB9000:IFX3=0THE
NJU=JU+1
1280 RETURN
1290 REM FUNCION POS
1300 REM
1310 X2=0
1320 IFX1+(LEN(X2$))-1)>LEN(X1$)THEN
RETURN
1330 IFMID$(X1$,X1,LEN(X2$))=X2$THE
NX2=X1:RETURN
1340 X1=X1+1:GOTO1320
9000 REM MOD
9010 REM
9020 REM DATOS DE ENTRADA :
9030 REM
9040 REM X1=DIVIDENDO
9050 REM X2=DIVISOR
9060 REM
9070 REM DATOS DE SALIDA
9080 REM
9090 REM X3=PARTE FRACCIONARIA
9100 REM X4=RESTO
9110 REM
```

```
9120 X3=INT(X1/X2)

9130 X4=X1-X3*X2

9140 X3=(X1/X2)-X3

9150 RETURN

READY.
```

PROGRAMA:11SOFTBASE3

```
1000 REM GREGORIANA

1010 REM CONVERSION DE JULIANA A GR
EGORIANA

1020 REM ---------------------------
--------

1030 REM ERNESTO MTNZ. DE CARVAJAL
HEDRICH

1040 REM 25/5/83

1050 REM ---------------------------
--------

1060 REM PARAMETROS DE ENTRADA : FE
CHA JULIANA EN VARIABLE JU

1070 REM PARAMETROS DE SALIDA   : FE
CHA GREGORIANA EN VARIABLE FE$

1080 REM ---------------------------
--------

1100 DIM ND(12)

1110 DATA 31,28,31,30,31,30,31,31,3
0,31,30,31

1130 FORI= 1 TO 12:READ ND(I):NEXT
I

1140 NM=0 : DI=0 : ME=0

1150 NM=NM+1 : ME=ME+1 : ND(2)=28
```

EJEMPLOS DE CONVERSIÓN DE FECHAS ENTRE LOS DOS FORMATOS

GREGORIANA		JULIANA
1/1/1980	<———>	1
29/2/1980	<———>	60
31/12/1980	<———>	366
31/12/1981	<———>	731
1/1/2000	<———>	7306
•		
•		
•		
31/12/9999	<———>	2929225

Y, para acabar con el tema de las fechas, ésta es una función para calcular el día de la semana de una fecha posterior al 1/1/1980 (se puede modificar fácilmente para cualquier otra base).

Utiliza tres de las funciones que os he definido hasta ahora: la POS, la de conversión a juliana y la MOD.

NOTA: En todas las funciones que os vaya definiendo, las sentencias de dimensionado y asignación de variables por medio de sentencias data, se han de trasladar a la zona del programa principal donde se inicialicen las variables del mismo, ya que de lo contrario al acceder por segunda vez a la rutina se produciría un error de redimensionado o de fin de sentencias data.

PROGRAMA:11SOFTBASE4

```
10 REM *** DIA SEMANA ***

15 REM ---------------------------
--------

20 REM

30 REM CALCULO DEL DIA DE SEMANA

40 REM

50 REM ERNESTO MTNZ. DE CARVAJAL H.

60 REM

65 REM ---------------------------
--------

70 DATA LUNES,MARTES,MIERCOLES,JUEV
ES,VIERNES,SABADO,DOMINGO

80 DATA 1

90 FORI=1TO7:READD$(I):NEXTI:READDB

95 REM ---------------------------
--------

100 INPUT "FECHA : ";FE$

110 GOSUB 1000

120 X1=JU:X2=7:GOSUB 9000

130 PRINT D$(X4+DB)

140 GOTO 100

150 REM ---------------------------
--------

1000 REM JULIANA

1010 REM CONVERSION GREGORIANA A JU
LIANA

1020 REM

1030 REM ERNESTO MTNZ. DE CARVAJAL
HEDRICH

1040 REM 25/05/83

1050 REM

1060 REM PARAMETROS DE ENTRADA : FE
CHA EN FORMATO DD/MM/AAAA EN VARIAB
LE FE$

1070 REM PARAMETROS DE SALIDA   : FE
CHA EN JULIANA EN VARIABLE JU

1080 REM

1090 DIM ND(12)

1100 DATA 31,28,31,30,31,30,31,31,3
0,31,30,31

1110 FOR I=1 TO 12:READ ND(I):NEXT
I
```

(termina en la pág. siguiente)

8

funciones de manejo de fechas (conclusión)

```
1120 X1$=FE$:X2$="/":X1=1:GOSUB 129
0:REM POS

1130 DI=VAL(LEFT$(FE$,X2-1))

1140 X3=X2+1:X1=X3:GOSUB 1290:REM P
OS

1150 ME=VAL(MID$(FE$,X3,(X2-X3)))

1160 AN=VAL(RIGHT$(FE$,4))

1170 JU=(AN-1980)*365

1180 IF ME=1 THEN 1200

1190 FOR I=1 TO ME-1:JU=JU+ND(I):NE
XT I

1200 JU=JU+DI

1210 IF AN-1980=0THEN1260

1220 FOR I=1980 TO AN-1

1230 X1=I:X2=100:GOSUB9000:IFX3=0TH
ENX2=500:GOSUB9000:IFX3=0THEN1250

1240 X1=I:X2=4:GOSUB9000:IFX3=0THEN
JU=JU+1

1250 NEXT I

1260 X1=AN:X2=100:GOSUB9000:IFX3=0T
HENX2=500:GOSUB9000:IFX3=0THEN1280

1270 IFME>2THENX1=AN:X2=4:GOSUB9000
:IFX3=0THENJU=JU+1

1280 RETURN

1290 REM FUNCION POS

1300 REM

1310 X2=0

1320 IFX1+(LEN(X2$))-1>LEN(X1$)THEN
RETURN

1330 IFMID$(X1$,X1,LEN(X2$))=X2$THE
NX2=X1:RETURN

1340 X1=X1+1:GOTO1320

9000 REM MOD

9010 REM

9020 REM DATOS DE ENTRADA :

9030 REM

9040 REM X1=DIVIDENDO

9050 REM X2=DIVISOR

9060 REM

9070 REM DATOS DE SALIDA

9080 REM

9090 REM X3=PARTE FRACCIONARIA

9100 REM X4=RESTO

9110 REM

9120 X3=INT(X1/X2)

9130 X4=X1-X3*X2

9140 X3=(X1/X2)-X3

9150 RETURN

READY.
```

clave para interpretar los listados de CLUB COMMODORE

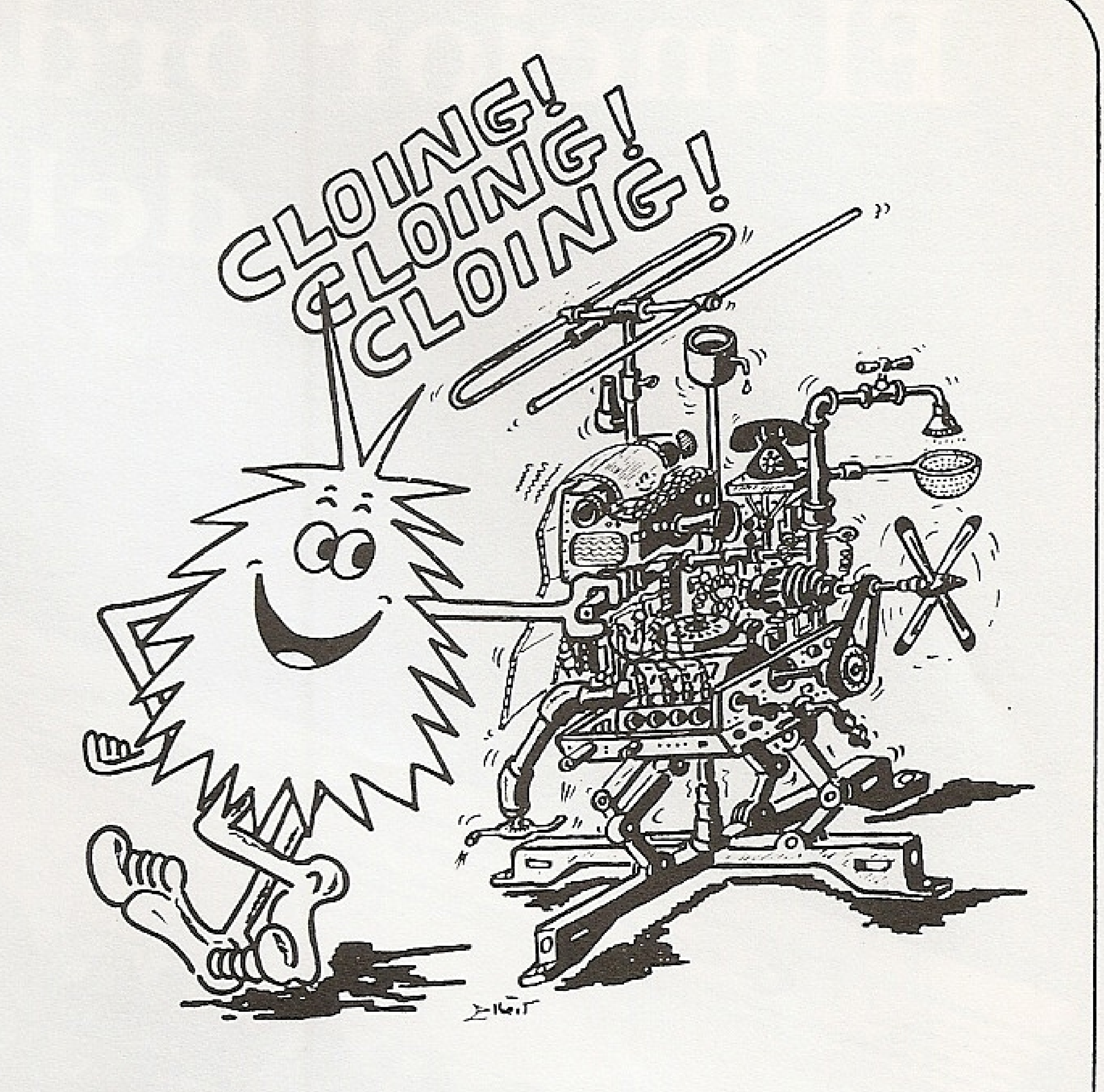
Todos los listados que se publican en esta Revista han sido ejecutados en el modelo correspondiente de la gama de ordenadores COMMODORE. Para facilitar la edición de los mismos en la Revista y para mejorar su legibilidad por parte del usuario, se les ha sometido a ciertas modificaciones mediante un programa escrito especialmente para ello. Para los programas destinados a los ordenadores VIC-20 y COMMODORE 64, en los que se usan frecuentemente las posibilidades gráficas del teclado, se han sustituido los símbolos gráficos que aparecen normalmente en los listados por una serie de letras entre corchetes [] que indican la secuencia de teclas que se deben pulsar para obtener el carácter deseado. A continuación se da una tabla para aclarar la interpretación de las indicaciones entre corchetes:

[CRSRD] = Tecla cursor hacia abajo (sin SHIFT)
[CRSRU] = Tecla cursor hacia arriba (con SHIFT)
[CRSRR] = Tecla cursor a la derecha (sin SHIFT)
[CRSRL] = Tecla cursor a la izquierda (con SHIFT)
[HOME] = Tecla CLR/HOME (sin SHIFT)
[CLR] = Tecla CLR/HOME (con SHIFT)

Las indicaciones [BLK] a [YEL] corresponden a la pulsación de las teclas de 1 a 8 junto a la tecla CTRL. Lo mismo sucede con [RVSON] y [RVSOF] respecto a la tecla CTRL y las teclas 9 y 0.

El resto de las indicaciones constan de la parte COMM o SHIF seguidas de una letra, número o símbolo — por ejemplo [COMM+] o [SHIFA] —. Esto indica que para obtener el gráfico necesario en el programa deben pulsarse simultáneamente las teclas COMMODORE (la que lleva el logotipo) o una de SHIFT y la tecla indicada por la letra, el número o el símbolo, en el ejemplo anterior: COMMODORE y + o SHIFT y A, respectivamente. ■

código máquina del 6502 (I)



por P. MASATS

Generalmente los ordenadores personales se pueden programar de forma inmediata en lo que se conoce como un lenguaje de ALTO NIVEL, del que el utilizado con más frecuencia es el BASIC. Lo de ALTO NIVEL no quiere decir que el lenguaje sea más complicado en su uso sino que se trata exactamente de lo contrario: es un lenguaje que se aproxima a la manera en que solemos plantear los cálculos en el mundo exterior al ordenador.

Para el usuario el hecho de trabajar con un lenguaje de ALTO NIVEL representa un gran ahorro de tiempo y de esfuerzo — particularmente cuando se está iniciando en la informática —. Ello le permite concentrarse en la tarea de captar los conceptos fundamentales sobre los que se basa el funcionamiento de los ordenadores. Pero (siempre hay un «pero») los lenguajes de ALTO NIVEL limitan de una forma u otra el funcionamiento del equipo, bien sea restringiendo el número de funciones que se pueden realizar, bien haciendo un uso poco eficiente de los recursos de memoria del equipo o realizando su trabajo a velocidades excesivamente bajas.

Para solucionar estos problemas se suele recurrir a lo que se conoce como CÓDIGO-MÁQUINA que no es más que el lenguaje de más bajo nivel que se puede utilizar con un microprocesador. Aquí tropezamos con un concepto que no hemos explicado: el del microprocesador; éste suele ser un circuito integrado que se encarga en el ordenador de realizar las tareas de cálculo, gestión de la memoria y manejo de los periféricos del sistema, es decir lo que también se conoce como CPU (Central Processing Unit o Unidad Central de Proceso); el funcionamiento de este circuito se programa en un lenguaje específico que se denomina CÓDIGO-MÁQUINA, generalmente con la ayuda de un compilador de ASSEMBLER, cuyo funcionamiento veremos más adelante.

Recapitulando: los ordenadores pueden ser programados en diferentes lenguajes, que se clasifican en diferentes niveles, dada su facilidad de programación o su versatilidad. Conforme sube el nivel aumenta la facilidad y disminuye su eficacia. Al lenguaje en el que se programa directamente el microprocesador que equipa el ordenador en el que trabajamos, se le llama CÓDIGO-MÁQUINA y es específico de la marca (a veces también del modelo) de circuito integrado que se utilice. Esto está expresado en términos generales y algo esquemáticos pero es válido para nuestro propósito que es el de explicar qué papel juega en informática personal el CÓDIGO-MÁQUINA.

La finalidad de esta serie de artículos es la de explicar el funcionamiento en CÓDIGO-MÁQUINA del microprocesador MCS6502 diseñado y fabricado por la compañía MOS TECHNOLOGY, que equipa a todos los modelos de ordenadores de COMMODORE, de la que MOS TECHNOLOGY es una empresa filial y a muchos otros del mercado de ordenadores personales, siendo este microprocesador uno de los más populares. De ello se deduce que el contenido de estos artículos será de interés para aquellos usuarios de ordenadores personales cuyas CPUs sean 6502 (APPLE, ACORN-ATOM, AIM-65, ATARI, etc...). Esta serie de artículos se han redactado con ejemplos realizados en los modelos VIC-20 y C-64 de COMMODORE, en cuyo caso el lector que posea otros equipos deberá tenerlo en cuenta y realizar las oportunas adaptaciones (que, como los ejemplos son muy generales, no presentarán grandes dificultades).

La idea general de esta serie de artículos — nos resistimos enérgicamente a llamarles cursillo — es la de hacer llegar al usuario audaz al «corazón» de su equipo. El aprendizaje del CÓDIGO-MÁQUINA tiene fama de ser difícil y complejo y sería engañar al lector caer en la tentación de negarlo aunque, en nuestra opinión, esto no es exacto: creemos que aprender a trabajar en CÓDIGO-MÁQUINA es simplemente laborioso por la sencilla razón de que nos movemos en el terreno más alejado de la lógica del «mundo exterior». Si el usuario está motivado por las mejoras en flexibilidad, potencia y rapidez que pone a su disposición el CÓDIGO-MÁQUINA no ha de tener dificultad en realizar el esfuerzo de aprender a utilizarlo. Otro argumento en favor de este «lenguaje» es el de que permite el contacto más «íntimo» posible entre el programador y su máquina facilitando el conocimiento exhaustivo del equipo que tenemos entre manos.

Hemos intentado que este texto sea accesible a cualquier usuario de ordenadores personales que esté familiarizado con el lenguaje BASIC, por ser el más extendido. Por ello explicaremos cada nuevo concepto según su orden de aparición e intentaremos no cansar al lector con áridas disquisiciones técnicas.

(continúa en la pág. 12)

El mejor ordenador personal del Mundo



El COMMODORE 64

Este es el nuevo ordenador personal COMMODORE 64. Un gigante de 40 cm, con un precio casi tan pequeño como su tamaño.

Nadie hasta ahora había logrado ofrecerle 64 K de memoria, 40 columnas en pantalla, 8 sprites y un sonido de auténtica maravilla por sólo 110.000,— ptas. Claro que tampoco todo el mundo es el líder mundial en microordenadores.

COMMODORE sabe perfectamente que para seguir siendo el número uno, tiene que estar constantemente en vanguardia. De calidad. De precios. De todo. Para ello investigamos constantemente.

Afortunadamente nuestra labor se ve plenamente recompensada cuando vemos, como lo demuestra el cuadro comparativo, que nuestro más directo competidor cuesta nada menos que un 100% más caro. Y ello sin reunir todos los adelantos técnicos del COMMODORE 64.

---

1. Capacidad total de memoria RAM de 64 K. Interpretador BASIC extendido y sistema operativo residentes en ROM.

2. Dotado del más potente chip sintetizador de sonido diseñado hasta hoy, el COMMODORE 64 ofrece 3 voces totalmente independientes con una gama de 9 octavas. El programa puede controlar la envolvente, la afinación y la forma de onda de cada voz,

convirtiendo al COMMODORE 64 en el mejor simulador de instrumentos.

3. Conectable directamente a toda una gama de periféricos, incluyendo unidad de discos, impresora de matriz de puntos o de margarita, plotter, comunicaciones locales y remotas..., y mucho más.

4. Pantalla de alta resolución en color con 320 × 200 puntos directamente direccionables. Capacidad en modo carácter de 25 líneas por 40 columnas.

5. El chip de video, único en su género, permite el uso de 8 «Sprites» (figuras móviles en alta resolución y color). Los «Sprites» pueden moverse independientemente por programa de «pixel» en «pixel».

6. A cada «Sprite» se le asigna por programa un nivel de prioridad en caso de cruce con otro, consiguiendo efectos tridimensionales, existiendo también detección automática de colisiones.

7. Teclado profesional con mayúsculas y minúsculas, más 62 caracteres gráficos, todos ellos disponibles en el teclado y visualizables en 16 colores, en forma normal o bien en video invertido.

8. Encontrará a su disposición una completa gama de programas profesionales, incluyendo proceso de textos, sistemas de información, modelos financieros, contabilidad y muchas más aplicaciones.

9. Están en fase de desarrollo asimismo otros lenguajes tales como LOGO, UCSD PASCAL, COMAL, ASSEMBLER, etc. Todos los programas existentes de la gama COMMODORE, desde el VIC-20 hasta los modelos CBM pueden ser adaptados fácilmente.

10. Posibilidad de inserción de cartuchos con programas grabados en ROM, tanto profesionales como para educación y ocio.

11. Opción de un segundo procesador Z-80 para trabajar con sistema operativo CP/M (R).

EL COMMODORE 64 Y SU MAS DIRECTO COMPETIDOR

OPCION DE BASE	COMMODORE 64	Más directo competidor
Precio	110.000,— ptas.	El doble
Memoria usuario	64 K	48 K
Teclado profesional	SI	SI
Teclado con caracteres gráficos	SI	NO
Minúsculas	SI	NO
Teclas de función	SI	NO
Máxima capacidad disco	170 K a 1 M	143 K
AUDIO		
Generador de sonido	SI	SI
Sintetizador de música	SI	NO
Salida HI-FI	SI	NO
VIDEO		
Salida monitor	SI	SI
Salida para TV	SI	EXTRA
PERIFERICOS		
Cassette	SI	SI
Periféricos inteligentes	SI	SI
Bus serie	SI	NO
SOFTWARE		
Opción CP/M (R)	SI	SI
Ranura cartucho externo	SI	NO

commodore
COMPUTER

PARA MAS INFORMACION
DEL COMMODORE 64,
LLAMAR O ESCRIBIR A:
MICROELECTRONICA Y CONTROL
c/ Taquígrafo Serra, 7, 5º. Barcelona-29
Tel. (93) 250 51 03
c/ Princesa, 47, 3º, G. Madrid-8
Tel. (91) 248 95 70

Nombre ....
Dirección ....
Tel. ....
Población ....

SLOGAN

código máquina...

(viene de la pág. 9)

AHORA ALGO DE HISTORIA

En principio, era INTEL
y, en principio, fue el 4004
(De la biblia apócrifa del programador)

Para entender el papel que juega nuestro protagonista (el 6502) en el mundo de la informática necesitamos realizar una zambullida — muy rápida y esquemática — en la historia de los microprocesadores de 8 bits.

Hacia 1970 la compañía norteamericana INTEL lanzó al mercado de componentes electrónicos un circuito integrado de un tipo que había de revolucionar la informática — y al resto del Mundo, de paso —. Se trataba del primer microprocesador y era conocido por 4004. Este circuito integrado podía realizar él solo las funciones de Unidad Central de Proceso (CPU) que, hasta entonces, se tenían que implementar en una placa de circuito impreso con varios chips y sus componentes discretos complementarios. Manejaba a través de sus instrucciones cantidades de cuatro bits (la instrucción POKE del BASIC de uno de estos equipos podría almacenar en memoria cantidades de 0 a 15 en vez de 0 a 255) lo cual puede parecer una seria limitación pero debe pensarse que el salto de 0 a 4 bits es infinitamente mayor que el salto de 4 a 8.

Como puede adivinarse fácilmente el 4004 fue todo un éxito, lo que animó a INTEL a diseñar el 8008 que, como es fácil de deducir, fue ya un microprocesador de 8 bits (como el 6502).

En 1973 lanzó al mercado una «segunda generación» del 8008: el 8080, que es esencialmente una versión mejorada de su «antepasado». Con más instrucciones, más modos de direccionamiento y mayor velocidad de trabajo pero esencialmente con la misma filosofía en cuanto a su manera de trabajar. Hasta que MOTOROLA presentó en 1974 su 6800, INTEL no tuvo ningún competidor.

MOTOROLA INC. (empresa norteamericana, por supuesto) se dio cuenta del tremendo mercado que se estaba abriendo para los microprocesadores y decidió meter su baza personal. Y fue su baza personal porque tenía — esencialmente — dos opciones: A) podía luchar con INTEL en su propio terreno, es decir, realizar un nuevo y mejorado 8080 (lo que hizo en 1976 ZILOG con su Z80); o bien, B) saltarse el precedente y diseñar un microprocesador «avanzado» y totalmente nuevo en sus líneas generales, con lo que había alguna posibilidad de situarse como mínimo al lado de INTEL en la tribuna de los líderes. MOTOROLA se decidió por la opción B) y puso manos a la obra.

El producto resultante, el microprocesador 6800, está organizado en torno a las líneas estructurales clásicas de los ordenadores, con los dispositivos de entrada/salida manejados en las operaciones de acceso como si fueran posiciones de memoria. Se simplifica así considerablemente el diseño de los sistemas.

NUESTRO PROTAGONISTA: EL MCS6502

El (breve, esperamos) resumen anterior sirve para situar el contexto en el que aparece el «chico» de nuestra personal película. Éste fue desarrollado por ocho ex-empleados de MOTOROLA que vieron que algunos pequeños cambios de estructura interna y de software podían dar lugar a un circuito del tipo 6800 con grandes posibilidades en el mercado. Para ello se unieron a una compañía de circuitos integrados para calculadoras llamada MOS TECHNOLOGY.

El equipo de diseño de MOS TECHNOLOGY tenía dos objetivos en mente cuando desarrolló el 6502: **bajo costo** (quizás a alguien le parezca triste, pero es una realidad que el precio es una importante característica técnica) y **altas prestaciones.** Como sea que existe una correlación directa entre el coste y el tamaño del chip (la pastilla de silicio que contiene los transistores y las resistencias que forman el microprocesador), decidieron reducir la complejidad del diseño del 6800 para minimizar la cantidad de silicio requerida. Otras decisiones de diseño incluyeron la eliminación de uno de los dos registros de trabajo del 6800 y sus buffers tri-estado de salida de direcciones. Además cambiaron el registro índice de 16 bits por dos de ocho de funcionamiento separado y descartaron algunas de las instrucciones menos utilizadas.

La eliminación de instrucciones dejó espacio para incluir en el 6500 trece modos de direccionamiento, siete más que en el 6800. Esto da al 6502 capacidades que sólo se encuentran en los grandes ordenadores. Además, el equipo de diseño cayó en la cuenta de que mientras los ordenadores trabajan con numeración binaria, el hombre es un animal que piensa en decimal, por lo que dotaron al 6502 de un modo de cálculo y de un bit de control que le permite trabajar en binario o decimal. Esto significa que el programador no necesita realizar un «ajuste a decimal» en las operaciones de suma y resta.

Desde el punto de vista eléctrico, el 6502 se beneficia de una tecnología que le confiere buenas características de conmutación, bajo consumo de potencia (250 mW contra 600 mW típicos en el 6800) y una buena inmunidad al ruido.

El 6502 es uno de los doce microprocesadores compatibles en software que forman la serie 6500 (o también denominada 650X) que presentó MOS TECHNOLOGY en 1975. A través de acuerdos de segundas fuentes — producción y comercialización bajo licencia de un producto por otra empresa diferente de la que lo ha desarrollado —, estos dispositivos se fabrican también por ROCKWELL INTERNATIONAL y SYNERTEK. Los doce microprocesadores que forman la serie 6500 (a veces se le denomina familia) tienen el mismo conjunto de instrucciones y la misma estructura interna en cuanto a programación (la misma arquitectura), variando solamente en cuanto a tamaño y opciones de hardware. Esta familia es una de las más populares desde su aparición, equipando una gran variedad de productos entre ellos la mayoría de los ordenadores personales.

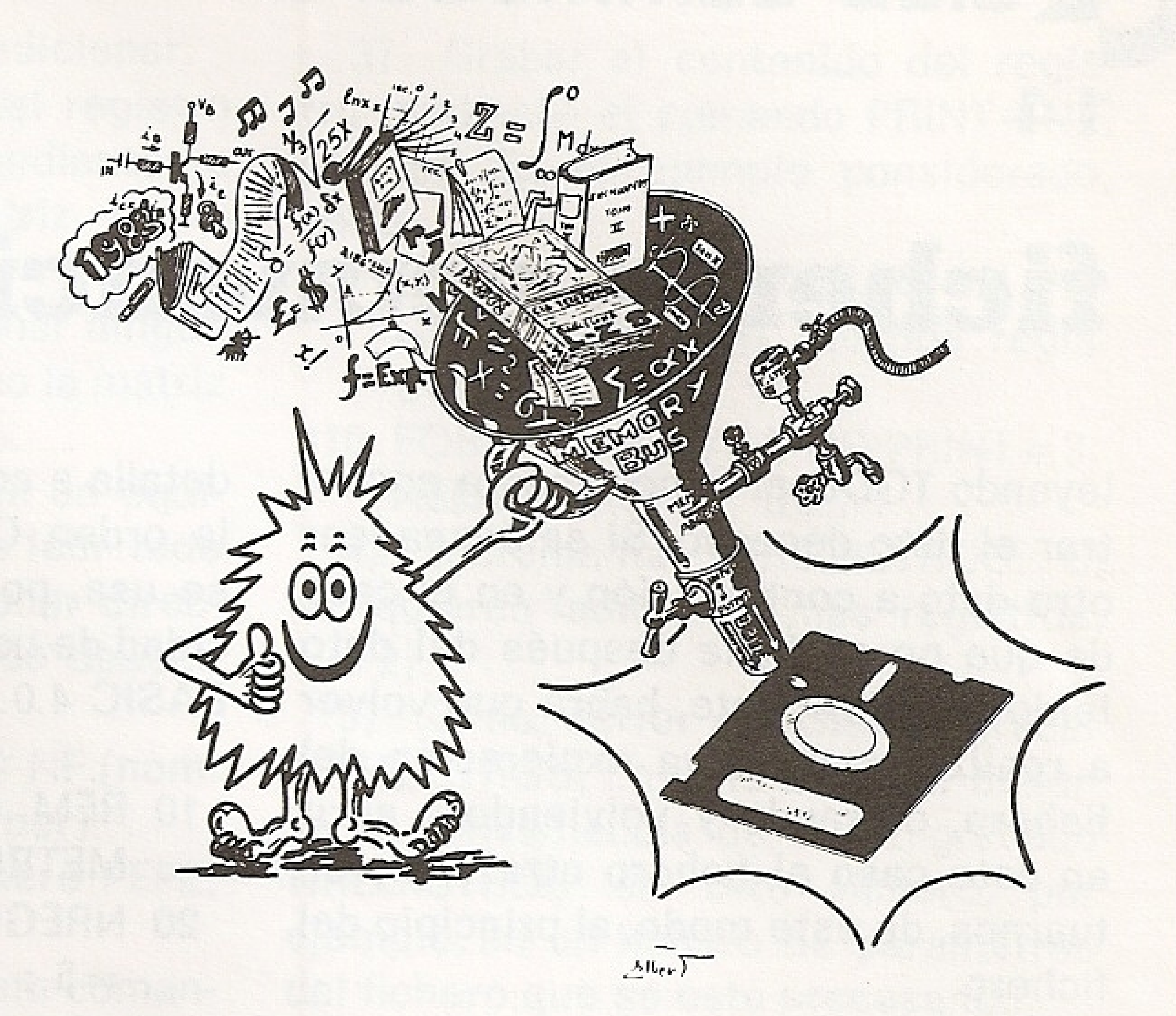
NOTA: En este resumen histórico se han manejado conceptos aún no explicados en cuanto a la estructura y característica de los microprocesadores (modos de direccionamiento, registros índice, etc...). Como la explicación de estos temas sobrepasa la finalidad de este apartado, se deja su análisis para los lugares en los que el contexto y el desarrollo de los conceptos lo requieran.

(Continuará)

FICHEROS CBM

FICHEROS SECUENCIALES

(continuación)



por **MANUEL AMADO**
(M.E.C. - SOFT)

C) LECTURA O ACCESO A LOS DATOS DEL FICHERO

Para leer los datos contenidos en un fichero secuencial, hay que tener previamente el fichero abierto en modo lectura. Por ejemplo, si queremos los datos del fichero PEPE, antes de ir a leer haremos:

OPEN 4,8,4,"PEPE,(SEQ),(R)"

En este caso, y tratándose de la LECTURA de un fichero SECUENCIAL, puede omitirse especificar el tipo de fichero (SEQ) y el modo de acceso al fichero, lectura (R)ead.

Una vez abierto el fichero, para poder leer los datos pueden usarse dos comandos BASIC distintos, INPUT # y GET #. Cada comando lee los datos de una forma determinada, y se usará uno u otro según cómo deseemos leer los datos y sobre todo según el formato en que estén grabados los datos.

Comando INPUT # NF, (lista variables):

Este comando se usa cuando los diferentes campos están grabados separados entre sí por CHR$(13). Si es una lista de variables, éstas se separarán mediante comas. Al ejecutarlo, asigna a cada variable todos los caracteres desde donde se encuentra el puntero de lectura del fichero, en el siguiente byte del último byte leído anteriormente, hasta que encuentra un CHR$(13). Este comando no se puede ejecutar en modo directo, sino dentro de una línea de programa.

Veamos un ejemplo. En el disco tenemos los datos de la tabla adjunta. (Donde cr=CHR$13)). Si ejecutamos la siguiente sentencia:

10 INPUT# 3,A$,B$,C$:?
A$" "B$" "C$

en pantalla tendremos:

PEPE SOL 12, o sea, A$="PEPE",B$="SOL",C$="12". Al ser el último campo de caracteres numéricos, también podríamos haber ejecutado:

10 INPUT# 3,A$,B$,C:?A$" "B$" "C
— — — PEPE SOL 12

Con lo cual, se observa que un campo NUMÉRICO puede ser asignado mediante INPUT # indistintamente a una variable numérica o alfanumérica. Pero si ejecutamos:

10 INPUT# 3,A,B,C, la CPU nos daría el error: ?FILE DATA ERROR IN 10, dado que estamos asignando un **dato alfanumérico** a una **variable numérica.**

Limitaciones del comando INPUT:

1. Si bien la longitud máxima de una variable alfanumérica BASIC es de 254 caracteres, el número máximo de caracteres de una variable a leer mediante el comando INPUT # es de 79 + CHR$(13) = 80 caracteres en total. Consecuentemente, ésta será la longitud máxima de un campo a leer por INPUT #.

2. La lectura y asignación de los datos a una variable finaliza al leer cualquier carácter ASCII especial, como pueden ser:

Retorno de carro, coma, etc. En este caso, el carácter especial o de control no es añadido a la variable.

Comando GET # NF, (lista variables):

La sintaxis y la forma de separar las variables es la misma que en el caso del comando INPUT #. Este comando lo que hace es leer y asignar a cada variable de la lista, UN SOLO carácter a cada una. Entonces diréis, ¿qué ventajas tiene sobre el INPUT #? Pues precisamente el hecho de que no tiene las limitaciones del comando anterior. Así, mediante un bucle FOR NEXT podemos leer campos de más de 80 caracteres, asignándolos a la variable correspondiente y leer también datos que contengan caracteres especiales.

Para finalizar, sólo indicaros que el comando GET # es extremadamente útil cuando se trata de explorar con detenimiento los datos que se hallan grabados en un fichero o hasta en un disquette determinado.

La única limitación de este comando es que en el caso de que lea un cero binario (CHR$(0)), dará cadena nula.

Para una mayor profundización sobre la problemática del uso de estos dos comandos, os recomiendo la lectura del artículo correspondiente a VENTANA CBM de Abril 1983, número 7, de nuestra estupenda Revista.

Acceso a los datos del fichero:

El fichero secuencial, si bien tiene la ventaja de que es el más rápido en grabación y en la lectura SECUENCIAL de los datos, mediante las instrucciones anteriormente vistas, presenta numerosos inconvenientes en el caso de que se desee acceder a un dato DETERMINADO. En este caso, una vez abierto el fichero, hay que ir

TABLA

CHAR	P	E	P	E	cr	S	O	L	cr		1	2	cr	eof
Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14

ficheros secuenciales

(viene de la pág. anterior)

leyendo TODO el fichero hasta encontrar el dato deseado. Si se desea leer otro dato a continuación y en el caso de que no se halle después del dato leído anteriormente, habrá que volver a realizar una nueva exploración del fichero, cerrando y volviendo a abrir en este caso el fichero otra vez y situarnos, de este modo, al principio del fichero.

Para ahorrarnos todos estos accesos continuados a disco, una solución podría ser, en el caso de consultas o lectura, leer de una vez todo el fichero al principio del proceso colocándolo en una matriz, generalmente bidimensional. Ello nos permitirá realizar las búsquedas en RAM, y el proceso de lectura será muchísimo más rápido que si se realizase sobre disco.

Veamos el siguiente ejemplo:

Sea el fichero "PEPE", de 200 registros y 5 campos/registro. Cada campo es alfanumérico de longitud máxima 30 bytes. Un programa de consulta, según lo expuesto en el párrafo anterior, y teniendo en cuenta de que los datos se han grabado separados entre sí por CHR$(13), podría ser el que se detalla a continuación. En vez de usar la orden OPEN para abrir el fichero, se usa, por ser de una mayor simplicidad de uso, la orden DOPEN # NF del BASIC 4.0.

```
10 REM DECLARACIÓN DE PARÁMETROS DEL FICHERO
20 NREGISTROS=200 : NCAMPOS =5 : DIM A$(200,5)
30 REM A$(I,J) : MATRIZ EN DONDE SE ALMACENA EL FICHERO EN RAM
40 DOPEN # 3,"PEPE",DO : FORI=1 TONREGISTROS : FOR J=1 TON CAMPOS
50 INPUT # 3,A$(I,J) : NEXT J,I : DCLOSE # 3
80 REM BUCLE DE LECTURA DE LOS REGISTROS DEL FICHERO
90 REM SE TOMA COMO CAMPO DE ACCESO EL CAMPO 1
100 INPUT "[HOME]CAMPO 1";C1$ : I=1
110 REM BUCLE DE BÚSQUEDA DEL PRIMER REGISTRO DE CAMPO 1 = C1$
120 IF I>NREGISTROS THEN GOSUB 200:GOTO100: REM C1$ NO EXISTE
125 IF LEFT$(A$(I,1),LEN(C1$))= C1$ THENGOSUB150:GOTO100: REM EX.
130 I=I+1 : GOTO120
150 REM C1$ ENCONTRADO : IMPRESIÓN REGISTRO
160 ?"[CRSR DOWN]":FOR J=1 TO NCAMPOS THEN GOSUB170: NEXT:RETURN
165 REM SUBRUTINA IMPRESIÓN CAMPOS ENCONTRADOS
170 ?"CAMPO";J;" :"A$(I,J):RETURN
200 ?"[CRSR DOWN]REGISTRO INEXISTENTE":FORJ=1TO800: NEXT
210 ?"[CLR/HOME]":RETURN
```

El programa finalizará respondiendo con RETURN solamente al INPUT de la línea 100.

D) MODIFICACIÓN, BORRADO Y ADICIÓN DE DATOS:

D.1 - Modificación:

Es un caso análogo al anterior pero con la diferencia de que, al encontrar el registro que se desea modificar, se ha de actualizar el contenido de los elementos de la matriz correspondientes. Y sobre todo, antes de salir del programa de modificación, y finalizado el proceso de modificación, ha de procederse a grabar otra vez el fichero en el disco, puesto que las modificaciones se han realizado sobre RAM, y hay que traspasar dicha modificación al disco. La rutina de grabación de la matriz podría ser la siguiente:

```
250 REM RUTINA DE GRABACIÓN DE LA MATRIZ EN EL FICHERO
255 DOPEN # 3,"@PEPE",DO,W : FORI =1TONREGISTROS : FORJ= 1TONCAMPOS
260 PRINT # 3,A3$(I,J);CHR$(13); : NEXTJ,I : DCLOSE # 3
```

D2. - Borrado de registros:

En este caso, el número de registros reales que tiene el fichero en cada momento, puesto que estamos considerando la posibilidad de eliminar — o como explicaremos en el siguiente apartado, de adicionar registros —, hay que tenerlo guardado en disco. Por ejemplo, como el primer

dato del fichero secuencial y, por tanto, el primer dato a leer. A continuación se leería el resto del fichero en una matriz.

La operatoria de trabajo, teniendo en cuenta el párrafo anterior, es la siguiente:

1) Buscar el registro que se desea borrar.

2) Si existe, eliminar dicho registro de la matriz, mediante un simple bucle de corrimiento de los elementos de índice mayor que el que se desea eliminar.

3) Decrementar NREGISTROS, y

4) Si no se desea borrar ningún registro más, grabar en disco la matriz y el NREGISTROS actualizado, por rutina 250.

D.3 - Adición de registros:

Para el caso de trabajar con BASIC 2 (VIC-20 y C-64), la adición de nuevos registros se realizará como sigue (una vez leído NREGISTROS):

1) Entrar el registro a adicionar.

2) Poner el contenido del registro en los elementos correspondientes a NREGISTROS+1 de la matriz. Incrementar NREGISTROS.

3) Si no se desea adicionar ningún registro más, grabar en disco la matriz y NREGISTROS actualizado.

Si estamos trabajando con un equipo con BASIC 4.0 en vez de leer todo el fichero, se pueden adicionar directamente los registros que se deseen usando:

— el comando APPEND#NF,(nom. fichero),d(n.drive),onu(per)

Donde en el caso del fichero PEPE, el formato sería:

APPEND#3, "PEPE",d0. Este comando es una forma especial de apertura del fichero, colocándose el puntero de acceso al final del fichero secuencial. De este modo podremos adicionar nuevos datos. En este caso sustituye al comando DOPEN#. Para escribir los datos a adicionar, se realizarán los pasos siguientes:

1) APPEND#3,"PEPE",d0 (Apertura en modo adición de datos).

2) Entrar el registro a adicionar.


3) Grabar el contenido del registro mediante el comando PRINT#NF, (datos). En el ejemplo considerado, sería:

```
300 REM RUTINA GRABACIÓN REGISTRO (en a$(j), campos registro))
310 FORJ=1TONCAMPOS:PRINT#3,
    A$(J);CHR$(13); :NEXT
```

4) Incrementar NREGISTROS. Si se quieren adicionar más registros, ir a 2).

5) Si no, cerrar el fichero y grabar NREGISTROS. En este caso se observa la conveniencia de tener grabado NREGISTROS en otro fichero, por ejemplo, en un fichero de parámetros del fichero que se está procesando.

Bien, y con esto doy por finalizado el tratamiento de los ficheros secuenciales. En el próximo artículo, os empezaré a hablar de la estructura del disquette y de la distribución de datos del mismo, así como de su filosofía de trabajo, para poder empezar a hablar próximamente de los ficheros de ACCESO DIRECTO a pista y sector del disco. ■



CBM - B700 (I)

los nuevos ordenadores de gestión serie 700



por **JORDI SASTRE** (M.E.C. - SOFT)

Iniciamos aquí una nueva sección de esta Revista, que titularemos «CBM - B 700», aunque más bien debería llamarse «Por fin !!!...»

Hace ya tiempo que Commodore había anunciado las nuevas series de ordenadores de gestión 500 y 700. Se han presentado prototipos de las mismas en todas las ferias habidas desde el verano del 82, en Londres («Commodore Computer Show»), Chicago («Summer Consumer Electronics Show»), Las Vegas («Winter C.E.S. Show»), Hannover, París («Sicob»), etcétera...

Acerca de estos nuevos aparatos se ha dicho de todo: rumores, comentarios, habladurías, opiniones a favor y en contra, que si son muy buenos, que si son muy malos. Todo provocado por la atmósfera de suspense que Commodore creó alrededor de ellos (a lo Alfred Hitchcock). Microelectrónica y Control presentó los nuevos modelos a su red de distribución en marzo del 83, y al público en este último SONIMAG. Entre ambas fechas ha habido un agotador trabajo del departamento de Software para conocer a fondo el aparato y desarrollar el MEC/DOS (potente sistema operativo, que pronto nacerá, para hacer las mil maravillas con la serie 700). En esta serie de artículos os vamos a contar todo (o casi todo) lo que sabemos:

Para empezar, hablemos de nombres: 700, 720, 500, serie B, serie BX, CBM-II, B128-80, B256-80, C-500... ¡Todos se refieren a los mismos aparatos! Nosotros los llamaremos serie B-700, que tendrá diferentes modelos: B-700, B-710, B-720, BX-720, etc., diferenciables por unas características concretas, que pueden resumirse en:

- 128K ó 256K de RAM libre usuario (ampliable a 896K).
- Opción con monitor de 80 × 25 (caracteres de 9 × 14).
- Opción con segundo microprocesador Intel 8088 o Zilog Z-80.
- Opción con dos drives incorporados de acceso DMA (680K).

Todos los modelos tendrán las siguientes características en común:

- Microprocesador 6509 con bus de direcciones de 16 bits y conmutador de bancos de 64K cada uno.
- Reloj de 2 MHz.
- Chip SID para sonido (con altavoz incorporado y audio-jack para sonido externo).
- Botón exterior de RESET.
- Bus IEEE-488.
- Interfaz RS-232-C.
- Port de cartridge (para cartuchos de 24K de ROM o RAM).
- Port de usuario de 8 bits.
- Pantalla de alta persistencia.
- Monitor orientable vertical y horizontalmente.
- Sistemas operativos alternativos (CP/M y MS/DOS).
- Posibilidad de trabajo en red.
- ROM de caracteres alternativa.
- Teclado separado (como la serie SK del 8000).
- 10 Teclas programables con hasta 20 funciones.
- Teclas de control de cursor separadas.
- Nuevo teclado numérico con ENTER, doble cero, tecla CE, ...
- Basic 4 ampliado con nuevos comandos: IF/THEN/ELSE, PRINT USING, Error Traping, DELETE, Conmutación de Bancos, etc...
- Posibilidad de carga de otros lenguajes por soft.
- Completísimo editor de pantalla (26 funciones mediante la tecla ESC).
- Teclas con hasta cuatro funciones (con SHIFT, CONTROL o ESC), etc...

Como veis, todo un ordenador. Su potencia no se puede determinar simplemente enumerando sus características sino comentándolas. A lo largo de varios meses os iremos dando la paliza sobre la serie 700. Hablaremos de su teclado, su editor de pantalla, sus bancos de memoria, sus ports, los sistemas operativos incorporables, sus nuevos comandos, el MEC/DOS, y un largo etcétera.

Empezaremos por:

EL TECLADO

Una de las mayores diferencias entre la serie 700 y los anteriores modelos de Commodore está en el teclado.

Dispone de 10 teclas programables con hasta 20 funciones (utilizando SHIFT). Cada tecla puede contener hasta 255 caracteres, aunque, combinadas, no pueden sobrepasar los 512. Al poner en marcha el ordenador, éstas vienen asignadas con contenidos como «LIST», «DIRECTORY», «PRINT», etc. Para cambiarlas se usa el coman-

(continúa en la pág. 19)

los nuevos ordenadores de gestión serie 700

(viene de la pág. 17)

do **'KEY n,string'**, donde 'n' es el número de función (1-20) y 'string' es un literal. Por ejemplo:

KEY 1,"RUN"+CHR$(13)

provocará que al pulsar la tecla F1 se ponga en marcha el programa existente en memoria.

Las teclas de control de cursor están juntas y hay una por cada movimiento: arriba, abajo, derecha e izquierda (no es preciso usar SHIFT).

También están agrupadas las teclas CLR/HOME, OFF/RVS, NORM/GRAPH (para poner la pantalla en modo Texto o modo Gráfico), y RUN/STOP (en un extremo para que no se pulse por error).

El teclado alfanumérico es muy similar al de la serie 8000 pero incorporando además la tecla CONTROL para acceder a los gráficos por lo que muchas teclas tienen tres funciones diferentes (además de la ESC que veremos luego).

El teclado numérico también está ampliado con la tecla ENTER (equivalente a RETURN), CE (cancela la última entrada numérica), ? equivalente a PRINT), doble cero (00) y los signos aritméticos: + — * /. Todo ello en un «keypad» aparte.

La tecla ESC (ubicada en el teclado alfanumérico), seguida de una letra de la A a la Z, habilita una serie de funciones de edición de pantalla, que detallamos a continuación:

Tecla	Función
A	Activa INSERCIÓN AUTOMÁTICA de caracteres.
B	Delimita el ÁNGULO INFERIOR derecho de la pantalla activa.
C	Cancela A (inserción automática).
D	BORRA LA LÍNEA donde se halla el cursor.
E	Pone el CURSOR FIJO (sin intermitencia).
F	Cancela E (pone el CURSOR INTERMITENTE).
G	Activa el aviso acústico del final de la línea (columna 72).
H	Cancela G (desactiva el aviso acústico).
I	INSERTA UNA LÍNEA en donde se halla el cursor.
J	Envía el CURSOR AL PRINCIPIO DE LA LÍNEA (extremo izquierdo).
K	Envía el CURSOR AL FINAL DE LA LÍNEA (extremo derecho).
L	Cancela M (ACTIVA SCROLL).
M	DESACTIVA SCROLL de la pantalla.
N	Cancela R (pone la PANTALLA NORMAL).
O	CANCELA los modos de INSERCIÓN, COMILLAS, y REVERSE.
P	BORRA LA LÍNEA desde el principio HASTA LA POSICIÓN DEL CURSOR.
Q	BORRA LA LÍNEA desde la posición del cursor HASTA EL FINAL.
R	Pone la PANTALLA EN REVERSE.
S	Cancela U (pone el CURSOR SÓLIDO).
T	Delimita el ÁNGULO SUPERIOR izquierdo de la pantalla activa.
U	Pone el CURSOR EN MODO SUBRAYADO.
V	Efectúa un SCROLL hacia ARRIBA de toda la pantalla.
W	Efectúa un SCROLL hacia ABAJO de toda la pantalla.
X	Cancela ESC (por si se ha pulsado por error).
Y	Cancela Z (activa los CARACTERES NORMALES).
Z	Activa los CARACTERES ALTERNATIVOS.

Estas funciones pueden emplearse tanto en modo directo como en modo programa: PRINT CHR$(27)+"Q" tiene el mismo efecto que pulsar ESC y Q.

(termina en la pág. siguiente)

los nuevos ordenadores...

(conclusión)

Finalmente, otra gran innovación en este editor es el «Wrap Bitmap», o «Control de Líneas Continuadas». Ello permite escribir líneas de programa que ocupen dos líneas de pantalla (160 caracteres), y no limitar la longitud de las líneas impresas en pantalla; es decir, al ejecutar:

FORT X = 1 TO 1000 : PRINT "A";
: NEXT

en la pantalla aparecerán 1000 caracteres "A" como en cualquier ordenador. Pero, se observará que las líneas que haya por debajo de la impresión se irán corriendo hacia abajo para no ser pisadas por las múltiples "A"s que se están imprimiendo; y que las trece líneas que ocupan las 1000 "A"s forman una sola. De manera que, por ejemplo, pulsando ESC Q (borrado hasta el final de la línea) en cualquiera de las líneas de "A"s, no se borrará hasta la columna 80 de la línea en curso sino hasta la columna 80 de la última de las líneas de "A"s.

Podríamos estar hablando eternamente de este editor de pantalla pero, por ahora, lo dejaremos, y el mes que viene comentaremos más cosas de la nueva serie 700. ¡Hasta entonces! ■

CLUB DE USUARIOS DE ORDENADORES COMMODORE

Sigue abierta la inscripción en el Club de Usuarios de Ordenadores Commodore. En él encontrarás información sobre revistas, libros, accesorios y software. Recibirás boletines con programas, trucos y técnicas de programación, POKES, mercado de ocasión, contactos... Al mismo tiempo tendrás acceso a nuestra biblioteca y programoteca, obtendrás descuentos en nuestros cursillos y en la compra de accesorios y software. Toda una serie de ventajas para que el trabajo con tu ordenador sea más rápido y efectivo. El precio de la inscripción anual es de 3.000 ptas., lo que te da derecho a recibir una cassette con varios programas muy útiles, nuestros boletines, y a participar en las actividades antes mencionadas. Si deseas más información, escribe a: Club de Usuarios de Ordenadores Commodore. Vía Augusta, 120. BARCELONA-6. Recibirás el número 0 de nuestro boletín y toda la información que solicites.

MARKETCLUB

● **ACCESORIOS VIC-20: Ampliación de memoria 16K más varios programas, 13.000. Módulo de expansión para 6 cartuchos, 10.000. Cartucho lenguaje FORTH y manual, 7.000. Las tres cosas sólo por 25.000. Jaime. Tel. 245 46 56. Barcelona.**

● **En Barcelona, clases de informática. PLAZAS LIMITADAS. Lenguaje BASIC. Prácticas con micro-ordenador VIC-20. Prof. E. Martínez de Carvajal. Información: Tel. (93) 345 10 00. Señorita María José (mañanas) o (93) 345 87 75. Sr. Martínez (fuera de horas de oficina).**

● **Vendo módulo para VIC-20 compuesto de: dos puertas de ocho entradas/salidas más dos de control y estado. Dos temporizadores programables. Posibilidad de interrupción IRQ-NMI. Con las correspondientes instrucciones. Razón: Luis Torrents. C/. Velázquez, 39. TERRASSA (Barcelona).**

● **Hago programas en BASIC COMMODORE bajo encargo. Vendo VIC-20, con 16 K de RAM, cartuchos de: Ayuda al Programador, Monitor Lenguaje Máquina, Superexpander + 3 K y tres juegos. Además se darán al comprador muchos programas hechos y comprados y 5 libros existentes o no en España sobre el VIC. Todo por 65.000 ptas. (vale más de 100.000). Por separado también. Llamar al 91 - 253 13 40. Horas comida y cena. Dirección: Francisco Gutiérrez. Santiago Rusiñol, 12. MADRID-3.**

● **Vendo aplicación de facturación con control de representantes, 9 listados, 6 ficheros, estadísticas, etc. Permite copias de seguridad. Configuración: VIC 8 K, disco e impresora, 40.000 pesetas. Escribir a Jaime Ameller Pons. General Mola, 15, 1º B. CALATAYUD (Zaragoza).**

● **Se ofrece 3008 + C2N de segunda mano en buenas condiciones; precio a convenir. Razón: Domingo Garrofé Trabal. Aragón, 386, 1º 1ª. Tel. 211 54 40. BARCELONA-9.**

● **Vendo interfaz y programa para RTTY y CW para el PET a 15 K. Rafael, EA3CGK. Avenida Barcelona, 21, A, 4º 2ª. IGUALADA (Barcelona).**

INFORMACIÓN PARA NUESTROS SUSCRIPTORES

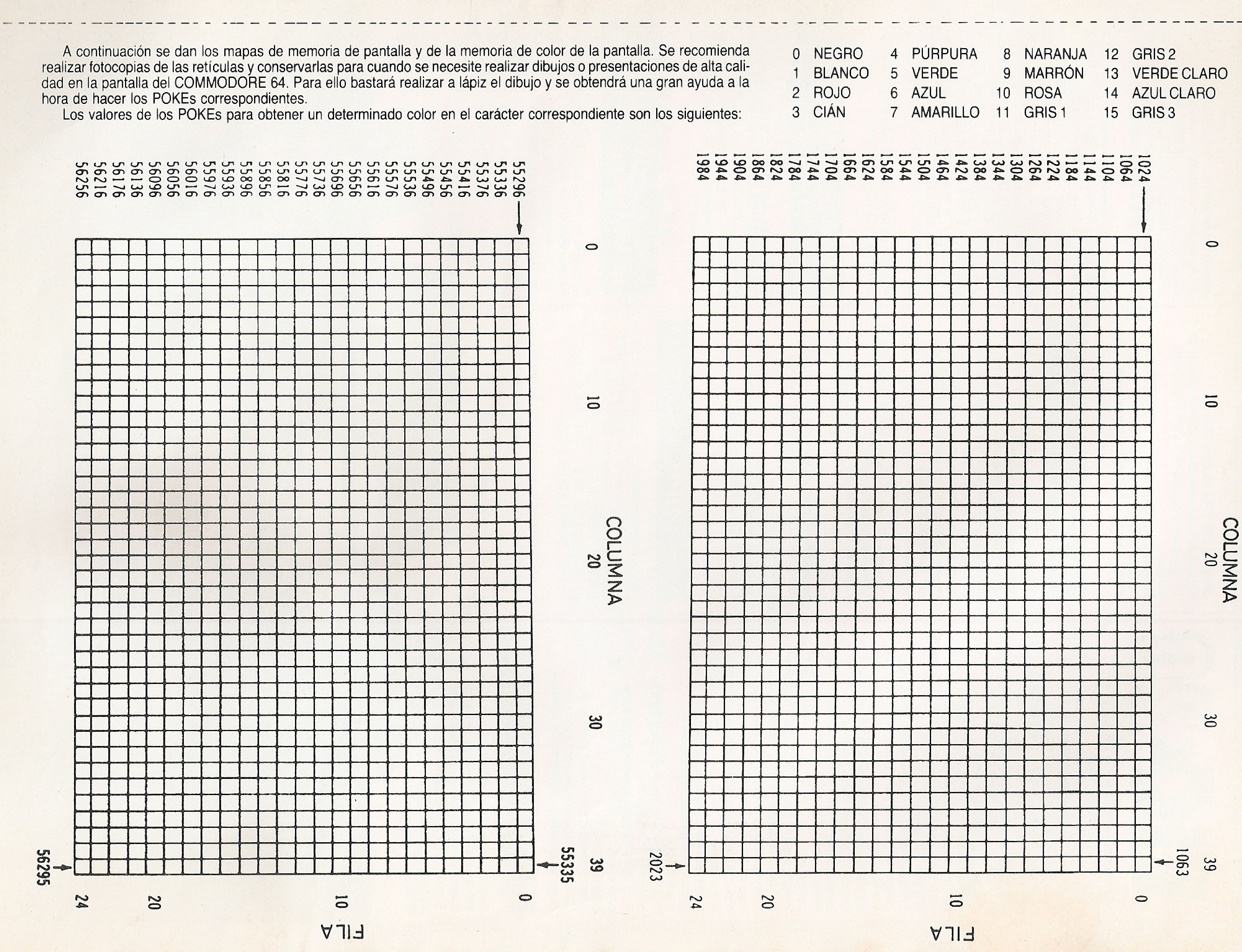
Hemos tenido noticia de que, durante los tres últimos meses, diversos suscriptores de nuestra Revista han sufrido anomalías en la recepción de sus ejemplares. Rogamos que cualquier irregularidad que observen en la entrega de los mismos nos la comuniquen de inmediato, para tratar de solucionarla en el plazo más rápido posible. Debemos hacer constar que, por nuestra parte, la entrega de la Revista para su distribución se realiza puntualmente, existiendo causas ajenas a nuestra Organización que pueden generar retrasos en la recepción. No obstante, lamentamos que se hayan producido anomalías e invitamos a quienes se hayan visto privados de algún ejemplar de su suscripción, a que nos lo indiquen.

MAPAS DE LAS MEMORIAS DE PANTALLA Y DE COLOR

A continuación se dan los mapas de memoria de pantalla y de la memoria de color de la pantalla. Se recomienda realizar fotocopias de las retículas y conservarlas para cuando se necesite realizar dibujos o presentaciones de alta calidad en la pantalla del COMMODORE 64. Para ello bastará realizar a lápiz el dibujo y se obtendrá una gran ayuda a la hora de hacer los POKEs correspondientes.

Los valores de los POKEs para obtener un determinado color en el carácter correspondiente son los siguientes:

0	NEGRO	4	PÚRPURA	8	NARANJA	12	GRIS 2
1	BLANCO	5	VERDE	9	MARRÓN	13	VERDE CLARO
2	ROJO	6	AZUL	10	ROSA	14	AZUL CLARO
3	CIÁN	7	AMARILLO	11	GRIS 1	15	GRIS 3



Depósito Legal Sep. B. 2133-1958 - Altés, s.l., Caballero 87, Barcelona

VIC-20

Microprocesador: 6502 de MOS TECHNOLOGY de 8 bits.
Memoria: 5 Kbytes de RAM ampliables a 32 K 20 Kbytes de ROM ampliables a 28 K
Pantalla: 23 lineas de 22 caracteres Modulador para conectar a un televisor normal. Salida para monitor de video.
Colores: 8 para el marco, 16 para el fondo de la pantalla y ocho para los caracteres individuales, video inverso.
Gráficos: Semi-gráficos por teclado y alta resolución por redefinición del generador de caracteres (situándolo en RAM). Definición de 176 por 184 puntos.
Teclado: Tipo QWERTY de 62 teclas más cuatro de función definibles por el usuario.
Sonido: Tres voces de tres octavas cada una decaladas una octava entre sí, resultando una extensión total de cinco octavas. Un generador de ruido aleatorio afinable para efectos especiales, un control general de volumen.
Programación: Lenguaje BASIC, intérprete residente en ROM de 8 K. Posibilidad de interceptar las funciones del Basic para crear nuevas instrucciones «a medida». El Basic del Vic es uno de los más rápidos actualmente en el mercado.
Complementos: Port de usuario de 8 bits entrada/salida más dos señales de sincronismo.
Bus de expansión para ampliaciones de memoria y periféricos.
Port de juegos con conexión para dos potenciómetros (paddles), y una palanca de juegos (joystick).
Almacenamiento de masa: Unidad de cassette C2N de diseño especial para registrar programas y datos (ficheros secuenciales).

VIC-1540 UNIDAD DE DISCO

Capacidad total: 174848 bytes por disco.
Secuencial: 168656 bytes por disco.
Entradas de directorio: 144 por disco.
Sectores por pista: De 17 a 21.
Bytes por sector: 256.
Pistas: 35.
Bloques: 683 (644 bloques libres).
Soportes de información: Discos estandar de 5 1/4 pulgadas, de una sola cara y densidad simple.
Sistema operativo: DOS de COMMODORE inteligente (tiene procesador propio y no ocupa memoria del ordenador central).

VIC-1515 IMPRESORA

Método de impresión: Matriz de 5×7 puntos, impacto por un solo martillo.
Modo caracteres: Mayúsculas y minúsculas, símbolos, números y caracteres gráficos del VIC-20.
Modo gráfico: Puntos direccionables (bit image). Siete puntos verticales por columna, 480 columna máximo.
Velocidad: 30 caracteres/segundo, de izquierda a derecha, unidireccional.
Caracteres/Linea: Máximo 80. (Posibilidad de impresión en doble ancho).
Espaciado entre lineas: 6 lineas/pulgada -modo caracteres, 9 lineas/pulgadas - modo gráfico.
Velocidad de salto de lineas: 5 saltos/seg. -modo caracteres, 7,5 saltos/seg. - modo gráfico.
Alimentación de papel: Arrastre por tractor.
Ancho de papel: Entre 4,5 y 8 pulgadas.
Copias: Original más dos copias.

CARTUCHOS

Ayuda programador: Este cartucho facilita la edición y depuración de programas en Basic. Instrucciones y comandos: RENUMBER, MERGE, FIND, CHANGE, DELETE, AUTO, TRACE, STEP, OFF, KEY, EDIT, PROG, DUMP, HELP y KILL.

Super expander: Intercepta el Basic del VIC permitiendo incrementar sus instrucciones y comandos en aplicaciones gráficas, de sonido y juegos. Instrucciones y comandos: KEY, GRAPHIC, COLOR, POINT, REGION, DRAW, CIRCLE, PAINT, CHAR, SCNCLR, SOUND, RGR, RCOLR, RDOT, RPOT, RPEN, RJOY y RSND.

Monitor de lenguaje máquina: Este monitor altamente sofisticado facilita enormemente la depuración de programas en lenguaje máquina, es ideal como complemento del Basic para redactar y poner en marcha rutinas de alta velocidad y manejo de datos en tiempo real. Instrucciones y comandos: ASSEMBLE, BREAKPOINT, DISASSEMBLE, ENABLE VIRTUAL ZERO PAGE, FILL MEMORY, GO, HUNT, INTERPRET, JUMP TO SUBROUTINE, LOAD, MEMORY, NUMBER, QUICK TRACE, REGISTERS, REMOVE BREACPOINTS, SAVE, TRANSFER, WALK y EXIT TO BASIC.

Además existen cartuchos de ampliación de memoria de 3, 8 y 16 Kbytes.

CURSO DE INTRODUCCION AL BASIC PARTE I:

En forma de libro se ha editado la primera parte de un curso de Basic que parte «de cero» y está basado en el VIC-20. Va acompañado de dos cassettes con programas y ejercicios para autocontrol de los progresos en el aprendizaje.

MODULO DE EXPANSION DE MEMORIA:

Acabado en metal de gran robustez, permite la conexión de un máximo de 6 cartuchos simultáneamente, aloja al VIC y al modulador de vídeo y permite colocar encima el televisor, tiene alojamiento para accesorios y asegura una óptima conexión del VIC a sus periféricos.









Taquígrafo Serra, 7 5.° Telf. 250 51 03. BARCELONA-29
Princesa, 47 3.° G. Telf. 248 95 70. MADRID-8